Werner Priess

Mecanismos de Escalonamento com Qualidade de Serviço em Redes com Tecnologia Bluetooth

Instituto de Matemática - Mestrado em Informática

Luci Pirmez

D.Sc. - COPPE/UFRJ - Brasil - 1996

José Ferreira de Rezende

Docteur, Université Pierre et Marie Curie - França - 1997

Rio de Janeiro

2003

Mecanismos de Escalonamento com Qualidade de Serviço em Redes com Tecnologia Bluetooth

Werner Priess

Dissertação (tese) submetida ao corpo docente do Instituto de Matemática da Universidade Federal do Rio de Janeiro - UFRJ, como parte dos requisitos necessários à obtenção do grau de Mestre.

Aprovada por:

Prof[a]. Luci Pirmez,
D.Sc. COPPE/UFRJ

Prof. José Ferreira de Rezende,
Dr. UPMC, França

Prof. Luiz Fernando Gomes Soares,
D.Sc. PUC/RJ

Prof. Marcelo Gonçalves Rubinstein,
D.Sc. COPPE/UFRJ

Prof. Oswaldo Vernet de Souza Pires,
D.Sc. COPPE/UFRJ

Rio de Janeiro - RJ
Janeiro de 2003

FICHA CATALOGRÁFICA

Priess, Werner.

Mecanismos de Escalonamento com Qualidade de Serviço em Redes com Tecnologia Bluetooth/ Werner Priess. Rio de Janeiro: UFRJ/IM/NCE, 2003

xiii, 98 p. 29,7 cm, il.

Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, IM/NCE, 2003

1. Redes Ad Hoc 2. Escalonamento com QoS 3. Bluetooth I. Título II. Tese (Mestr. UFRJ/IM/NCE)

A meus pais e meu irmão

AGRADECIMENTOS

A meus pais, que estiveram sempre presentes, propiciando-me todas as condições para o meu crescimento (em todos os sentidos).

A meu irmão, que, nos momentos de cansaço, sempre arrumava alguma "brincadeira" para me animar.

Aos amigos que fiz no NCE: Flávia, Reinaldo, Renata, Ana Paula, Alexandre, Roberta, César, Sidney, Eugênio, Denise, Patrícia, Roberto, Rosane, Noel e Leonardo. Espero não ter esquecido alguém.

Ao amigo Coelho, que realizou sua pesquisa concomitantemente à minha e com quem muitas idéias troquei.

Ao apoio recebido pelo pessoal do IME e, particularmente, aos professores Cecílio (grande amigo), Vidal (supervisor acadêmico) e Ana Maria Moura (outrora orientadora, agora uma das responsáveis pela realização deste trabalho de pesquisa no NCE).

A meus orientadores (e agora também amigos), Luci e Rezende, que sempre tiveram paciência para me ouvir e para me indicar o caminho a seguir, e ao Rust, que deu início ao trabalho.

A meus professores no NCE, pelo conhecimento proporcionado e ao pessoal do GTA, pela ajuda prestada.

Ao CNPq e à FAPERJ, pelo financiamento à pesquisa.

Resumo da Tese apresentada ao IM/NCE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

Mecanismos de Escalonamento com Qualidade de Serviço em Redes com Tecnologia Bluetooth

Werner Priess

Janeiro/2002

Orientadores: Prof^a.Luci Pirmez
Prof. José Ferreira de Rezende

Departamento: Informática

Bluetooth é uma tecnologia de rádio para comunicações de curto alcance, com baixo consumo de energia e baixo custo, favorecendo o seu emprego na implementação de redes *ad hoc*. A especificação da tecnologia é recente e ainda apresenta questões em aberto, incluindo o tópico de mecanismos de escalonamento das estações. Este trabalho de pesquisa propõe dois algoritmos de escalonamento: um para o escalonamento intrapicorrede, chamado DRR-QoS (*Deficit Round Robin with Quality of Service*) e outro para o escalonamento interpicorrede, denominado AISA (*Adaptive Interpiconet Scheduling Algorithm*). O DRR-QoS visa permitir o compartilhamento do enlace Bluetooth entre o tráfego de melhor esforço e o tráfego com requisito de retardo limitado, atendendo às necessidades de cada um. Assim, evita-se a criação de enlaces separados para os dois tipos de tráfego, conforme definido atualmente na especificação, otimizando-se a ocupação do canal. Já o AISA é um algoritmo adaptativo e parametrizável. A primeira característica faz com que as pontes se ajustem às condições do tráfego. A segunda possibilita que uma determinada métrica de desempenho seja priorizada. Estas duas características, juntas, permitem que o AISA seja empregado, com êxito, em cenários diferentes. As duas propostas foram testadas através de simulações, realizadas com a ferramenta BlueNetS. A ferramenta foi desenvolvida ao longo deste trabalho e implementa as principais funcionalidades das camadas física e de enlace Bluetooth. As duas propostas de algoritmos de escalonamento e a ferramenta são as principais contribuições deste trabalho.

Abstract of Thesis presented to IM/NCE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

Scheduling Mechanisms with Quality of Service for Bluetooth Networks

Werner Priess

Janeiro/2002

Advisors: Luci Pirmez
José Ferreira de Rezende

Department: Informatics

Bluetooth is a low power and low cost short-range radio technology that may be used for ad hoc networking. The technology specification is recent and it still has open issues, including scheduling mechanisms. This work proposes two scheduling algorithms: one for intrapiconet scheduling, called DRR-QoS (Deficit Round Robin with Quality of Service) and the other for interpiconet scheduling, called AISA (Adaptable Interpiconet Scheduling Algorithm). DRR-QoS allows best effort traffic and delay-sensitive traffic to share the same Bluetooth link, while respecting each one's needs. Thus, we avoid creating separate links for each traffic category, as defined currently in the specification, optimizing the channel occupation. AISA has two major characteristics: (1) it adapts to varying network traffic conditions and (2) it allows performance metric optimization through parametrization. These features, together, allow its use for different kinds of scenarios. Both proposals were simulated with the BlueNetS simulation tool. The tool was developed throughout this research work and it implements the main Bluetooth physical and link layer functionalities. The proposed scheduling algorithms and the BlueNetS tool are the most important contributions of this work.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Uma rede móvel *ad hoc*, conceituada de maneira informal, é aquela em que as estações se comunicam, automaticamente, conforme o seu interesse, sem a necessidade de uma prévia infra-estrutura de rede ou de alguma estação de rede fixa. Existem diversas aplicações para estas redes, desde as atividades cotidianas até o seu emprego em atividades militares. Foi justamente o emprego militar que impulsionou o estudo das redes *ad hoc*. Elas podem ser empregadas, por exemplo, em campos de batalha, em território desconhecido, onde a existência de uma rede infra-estruturada é improvável e onde há a necessidade do rápido estabelecimento das comunicações. As redes *ad hoc* também podem ser estabelecidas como solução de emergência em locais onde a infra-estrutura de comunicações tenha sido atingida por desastres naturais.

As redes de sensores sem fio constituem outra aplicação recente para as redes *ad hoc*. As redes de sensores são compostas de algumas dezenas até milhares de pequenos dispositivos, de baixa potência, com a capacidade de monitorar um ambiente ou equipamento e comunicar-se com outros elementos da rede. Existem muitos cenários onde as redes de sensores podem ser aplicadas: segurança de instalações, monitoramento das condições climáticas, detecção de falhas em equipamentos, entre outros.

As redes *ad hoc* também formam a base do paradigma da computação ubíqua, na qual pequenos dispositivos, integrados às pessoas e ao ambiente, comunicam-se de forma espontânea, sem a necessidade da intervenção do usuário. Ligado à computação ubíqua está o conceito de rede pessoal (PAN - *personal area network*), uma rede de curto alcance, englobando objetos usualmente associados a uma pessoa. O relógio

de pulso, o telefone celular, o PDA (*personal digital assistant*), a câmera digital e o computador portátil são exemplos de equipamentos que podem compor uma rede pessoal. Conforme algumas pessoas se encontram para participar de uma reunião, suas redes pessoais podem se conectar automaticamente, permitindo a troca de informações entre os participantes.

Diversas tecnologias de transmissão suportam a formação de redes *ad hoc*. As mais conhecidas são: IEEE 802.11 e variantes [1], HiperLAN [2], HomeRF SWAP [3] e Bluetooth [4]. As duas primeiras são normalmente empregadas na formação de WLANs (*wireless local area networks*) e as últimas, na formação de WPANs (*wireless PANs*). A diferença básica entre a WLAN e a WPAN é o alcance da rede. A WLAN possui um alcance na ordem de uma centena de metros, enquanto a WPAN limita-se a algumas dezenas de metros. Cada tecnologia foi especificada para suprir uma determinada necessidade do mercado, mas atualmente têm sido empregadas de forma concorrente.

Considerando os cenários citados, os módulos de rádio, embutidos nos equipamentos, devem satisfazer os seguintes requisitos:

- baixo custo – necessário para popularizar a tecnologia e permitir que a mesma seja empregada em dispositivos simples sem aumento de custo;
- tamanho reduzido – fundamental para que possam ser embutidos em dispositivos pequenos como micro-sensores, fones de ouvido e *mice*, sem aumento de volume;
- baixo consumo de energia – pequenos dispositivos não tem espaço físico para incorporar baterias de grande capacidade e, muitas vezes, a troca delas é inviável, ficando o equipamento inutilizado após o término da energia.

Dentre as principais tecnologias, Bluetooth [4, 5], que forma a base do recentemente aprovado padrão IEEE 802.15.1 [6] para redes pessoais sem fio (WPANs), mostra-se a mais promissora no tangente aos requisitos supracitados. O custo do chip Bluetooth está baixando, suas dimensões estão em torno de $25mm^2$ de área por 1mm de espessura e sua potência de transmissão é de cerca de 1mW, para distâncias em torno de 10m. Sua popularidade está crescendo e já estão disponíveis, comercialmente, diversos equipamentos integrados à tecnologia de rádio Bluetooth como, por exemplo, telefones celulares, fones de ouvido, computadores portáteis, PDAs, impressoras, *mice*, câmeras digitais, filmadoras, entre outros.

A estrutura básica para a formação de redes *ad hoc* com Bluetooth é a picorrede (do inglês *piconet*). A picorrede é uma rede composta por até oito dispositivos, onde um deles assume o papel de mestre da rede e todos os outros (os escravos) se comunicam somente com o mestre, formando uma topologia em estrela. A dimensão de oito dispositivos é suficiente para agrupar um microcomputador com todos os seus periféricos ou os dispositivos Bluetooth transportados por uma pessoa. Visando aumentar a quantidade de estações da rede e estender o seu alcance, surgiu o conceito de *scatternet*, na qual várias picorredes podem ser interligadas através de uma ou mais estações (aqui denominadas pontes) pertencentes, alternadamente, a múltiplas picorredes.

A especificação Bluetooth é recente, e ainda há questões em aberto, carecendo de padronização. Por exemplo, as *scatternets* foram definidas, mas os algoritmos necessários para seu funcionamento não foram especificados. Precisam ser estabelecidos mecanismos para a formação da *scatternet* e para a comunicação entre estações pertencentes a picorredes distintas. Finalmente, também existem questões relacionadas ao escalonamento em picorredes e *scatternets*.

1.2 Objetivos

Esta dissertação concentra-se na parte relacionada ao escalonamento em Bluetooth, o qual pode ser dividido em: intrapicorrede (*intrapiconet*) e interpicorrede (*interpiconet*). O intrapicorrede é escalonamento dos escravos, realizado pelo mestre dentro de uma picorrede. O escalonamento interpicorrede é o modo como uma estação pertencente a múltiplas picorredes divide seu tempo entre elas. Esta divisão ao longo do tempo faz-se necessária, pois uma estação, contendo uma interface Bluetooth, não pode estar presente em várias picorredes simultaneamente.

O estudo de mecanismos de escalonamento não é recente [7]. Entretanto, os resultados obtidos até então precisam de adaptações para que possam ser empregados ao Bluetooth, devido a particularidades dessa tecnologia, que tornam o seu escalonamento diferente do tradicional. Por isso, nos últimos anos, surgiram algumas propostas de escalonadores intrapicorrede e, mais recentemente, interpicorrede.

Basicamente, as propostas de escalonadores intrapicorrede [8, 9, 10, 11] buscam,

através de mecanismos adaptativos, otimizar a ocupação do canal, evitando o desperdício de banda. O mestre dedica maior parte do tempo aos escravos com maior tráfego sem, contudo, deixar de escalonar o restante dos escravos. Como a especificação Bluetooth prevê garantias específicas para o tráfego com requisito de retardo limitado como, por exemplo, o tráfego de voz, não houve grande preocupação em se limitar rigidamente o retardo dos tráfegos. Entretanto, as garantias citadas são obtidas ao custo de uma grande perda na vazão agregada da rede [10].

Os escalonadores interpicorrede visam dividir no tempo, de forma eficiente, a presença de uma estação (ponte) nas picorredes de que participa. Por razões que serão abordadas no Capítulo 3, essa estação não deve ficar alternando entre picorredes com muita freqüência, e os mestres das picorredes devem estar cientes dessa alternância. Novamente, a vazão agregada da rede constitui a principal métrica das pesquisas [12, 13, 14, 15, 16]. Entretanto, com a possibilidade dos tráfegos percorrerem múltiplos saltos, há uma preocupação maior, por parte dos autores, com o monitoramento do retardo. Além disso, o consumo de energia das pontes tende a ser maior que o dos escravos dedicados a uma única picorrede, criando outro fator de preocupação para o escalonamento.

O primeiro objetivo desta dissertação é o estudo dos mecanismos de escalonamento intrapicorrede existentes na literatura e a proposta de um mecanismo alternativo que permita a coexistência de tráfego de melhor esforço e tráfego com requisito de retardo limitado, maximizando a ocupação do canal, sem extrapolar o referido limite de retardo.

O segundo objetivo desta dissertação é a proposta de um mecanismo de escalonamento interpicorrede, adaptativo e parametrizável, a ser aplicado somente às estações pontes. A parametrização permite que, conhecidos os cenários, as redes sejam configuradas para favorecer, primordialmente, a uma métrica de desempenho pré-determinada como a vazão agregada, o retardo dos pacotes ou a economia de energia. A parametrização não interfere na manutenção da justiça entre os tráfegos com características semelhantes. O fato de ser empregado somente nas pontes evita grandes mudanças na especificação Bluetooth, tornando mais fácil sua implementação.

Para testar o funcionamento dos algoritmos propostos, foi desenvolvida a ferramenta de simulação BlueNetS (*Bluetooth Network Simulator*) [17], baseada no simulador

de redes NS-2 [18]. A ferramenta implementa as principais funcionalidades das camadas física e de enlace da pilha Bluetooth. Com estas funcionalidades, é possível estudar, além dos algoritmos propostos, diversos mecanismos relacionados às camadas superiores da pilha de protocolos como, por exemplo, o roteamento de pacotes, o comportamento do protocolo TCP e a descoberta de serviços.

1.3 Organização da Dissertação

A dissertação está dividida em seis capítulos, já incluída esta introdução. Os três primeiros capítulos abrangem os aspectos teóricos da dissertação. Os capítulos restantes exploram a proposta de escalonamento, sua implementação e os resultados. A divisão detalhada é descrita a seguir.

- Capítulo 1 – Refere-se a esta introdução
- Capítulo 2 – Descreve os conceitos básicos para o entendimento da dissertação. A primeira parte aborda, de maneira genérica, as redes móveis sem fio e o conceito de qualidade de serviço nessas redes. O restante do capítulo é destinado aos conceitos relacionados à tecnologia Bluetooth, incluindo a descrição de sua pilha de protocolos, seu funcionamento e os principais trabalhos relacionados.
- Capítulo 3 – Continua a explorar o Bluetooth, porém, com o enfoque nas *scatternets*. Essas redes apresentam questões em aberto suficientes para motivar sua separação do capítulo anterior. Além disso, esta topologia de rede é a base do desenvolvimento da proposta apresentada.
- Capítulo 4 – Explica os mecanismos de escalonamento intrapicorrede e interpicorrede propostos na dissertação. O escalonamento interpicorrede é explorado com mais detalhes, devido aos desafios ainda existentes no assunto e à sua importância para o êxito da implementação de redes *ad hoc* com Bluetooth.
- Capítulo 5 – Apresenta a ferramenta BlueNetS e os cenários de simulação e analisa os principais resultados das simulações obtidos em cada cenário.
- Capítulo 6 – Conclui a dissertação, destacando as vantagens e as desvantagens dos mecanismos propostos. Também apresenta, a partir da experiência adquirida, algumas previsões sobre o futuro da tecnologia Bluetooth. Finalmente, sugere propostas de continuação do trabalho, possibilitando a ampliação do conhecimento na área e a agregação de novas funcionalidades ao Bluetooth.

Capítulo 2

Conceitos Básicos

2.1 Introdução

Este capítulo apresenta diversos conceitos importantes para o correto entendimento desta dissertação. Os conceitos são abordados, inicialmente, de forma genérica, destacando-se, posteriormente, os de maior relevância.

A Seção 2.2 apresenta o conceito de redes móveis sem fio, classificando-as em duas categorias: redes infra-estruturadas e *ad hoc*. A seção seguinte fornece o conceito de Qualidade de Serviço (QoS) e, posteriormente, aborda novos parâmetros de qualidade, específicos de cenários de mobilidade das estações, mostrando as dificuldades em se atendê-los nesses ambientes. A Seção 2.4 descreve o Bluetooth, uma das tecnologias que permite a conectividade entre dispositivos móveis sem fio e que constitui a base desta dissertação. Finalmente, a Seção 2.5 apresenta um resumo do capítulo.

2.2 Redes Móveis sem Fio

Uma rede de computadores foi definida por Soares *et al.* [19] como sendo formada por um conjunto de módulos processadores (por exemplo, microcomputadores) capazes de trocar informações e compartilhar recursos, interligados por um sistema de comunicação. O sistema de comunicação é constituído de um arranjo topológico conectando os vários módulos processadores em um meio de transmissão e de um conjunto de protocolos. No caso de uma rede sem fio, o meio de transmissão é o ar. Finalmente, para ser considerada móvel, os módulos processadores devem estar aptos a se deslocar durante a comunicação, sem que esta seja interrompida.

O avanço tecnológico nas áreas de eletrônica e telecomunicações propiciou o surgimento e a popularização de dispositivos computacionais e de comunicação sem fio como telefones celulares, *personal digital assistants* (PDAs) e microcomputadores portáteis. O desenvolvimento das redes móveis sem fio ampliou o alcance das redes fixas. Além disso, as redes móveis atendem a uma série de novas aplicações, inviáveis para as redes fixas, devido a sua topologia estática. Podem ser citadas, por exemplo, o estabelecimento de redes residenciais, o acesso a serviços durante o deslocamento em trens ou aviões, os serviços via satélite e o paradigma da computação ubíqua. Essa última tem sido muito pesquisada, e seu conceito foi introduzido por Mark Weiser [20].

2.2.1 Redes Infra-estruturadas

Basicamente, as redes móveis sem fio podem ser divididas em redes *ad hoc* e redes infra-estruturadas. Nas redes infra-estruturadas, existe ao menos uma estação fixa, centralizadora, conhecida por estação-base ou ponto de acesso. Não há comunicação direta entre as estações móveis. Toda comunicação é intermediada pela estação-base. Como as estações-base têm alcance limitado, normalmente há várias estações-base interligadas, garantindo uma maior área de cobertura. A estação-base também pode estar conectada a outras redes como, por exemplo, a Internet, permitindo a comunicação entre estações móveis e fixas.

As redes de telefonia celular são exemplos de redes móveis sem fio infra-estruturadas. A célula é a região de cobertura de uma estação-base. Para se comunicar com qualquer outra estação, uma estação móvel transmite a informação para a estação-base de sua célula, e esta verifica se o destino se encontra na mesma célula, em outra célula ou na parte infra-estruturada da rede, fazendo o encaminhamento correspondente. Este tipo de rede não é abordado na dissertação.

2.2.2 Redes *Ad Hoc*

A rede *ad hoc* é definida pelo grupo de trabalho IETF MANET (*Mobile Ad hoc Networks*) [21] como um sistema autônomo de roteadores móveis (e equipamentos associados) conectados por enlaces sem fio. Os roteadores movem-se e comunicam-se

livremente e, por isso, a topologia da rede tende a mudar com freqüência e de forma imprevisível. As estações (ou roteadores) não precisam ser iguais, podendo ter diferentes alcances, capacidades de processamento e velocidades. Entretanto, todas devem estar aptas a executar as mesmas tarefas na rede, como o roteamento de pacotes para a comunicação com múltiplos saltos.

A independência de infra-estrutura e a liberdade de movimentação das estações possibilitam a rápida instalação de uma rede *ad hoc*, bem como sua reconfiguração automática em caso de falhas de rota. Por outro lado, essas vantagens são acompanhadas de alguns problemas. A mobilidade das estações causa muitas mudanças de rotas e perdas de pacotes, além de dificultar sua localização. Estações posicionadas no centro da rede tendem a gastar muita energia roteando pacotes, o que é inconveniente, já que, normalmente, possuem carga de bateria limitada. Essas limitações são abordadas com mais detalhes na próxima seção.

Os padrões mais conhecidos, aplicáveis como tecnologias de transmissão em redes *ad hoc*, são: Bluetooth [4], IEEE 802.11 e suas variantes [1], HiperLAN [2] e HomeRF SWAP[3]. Cada um foi especificado para suprir uma determinada necessidade do mercado, mas atualmente têm sido empregados de forma concorrente. A tecnologia Bluetooth, que forma a base do recentemente aprovado padrão IEEE 802.15.1 para redes sem fio pessoais (WPANs - *wireless personal area networks*) [6], foi a tecnologia empregada no desenvolvimento desta dissertação e será discutida na Seção 2.4 e ao longo do Capítulo 3.

2.3 Qualidade de Serviço em Redes Móveis sem Fio

A noção de qualidade de serviço (QoS) não é recente. Entretanto, não há um consenso com relação à sua definição. No padrão X.902 *International Telecommunication Unit* (ITU) [22], QoS é o conjunto de requisitos de qualidade relacionados ao comportamento coletivo de um ou mais objetos. No artigo de Vogel *et al.* [23], há uma definição de QoS para aplicações de tempo real: o conjunto das características quantitativas e qualitativas de um sistema multimídia distribuído, as quais são necessárias para se atingir a funcionalidade requerida por uma aplicação.

O fornecimento de QoS às aplicações está fundamentado na disponibilidade de recursos do sistema de comunicação, do qual fazem parte os roteadores, as estações e o meio de transmissão. Nesse nível, a QoS é especificada através de parâmetros como largura de banda solicitada, retardo de propagação, variação máxima do retardo, taxa de erros na transmissão, tempo de resposta a solicitações, entre outros. Eles são chamados de parâmetros de QoS baseados na tecnologia [24]. O surgimento das aplicações multimídia e dos sistemas multimídia distribuídos aumentou a demanda por recursos dos sistemas de comunicação e criou o conceito de requisitos de QoS em nível de usuário. Esses requisitos envolvem a qualidade de percepção das mídias por parte do usuário e são representados por parâmetros como resolução da imagem, taxa de amostragem, qualidade de áudio e sincronização entre áudio e vídeo. Um dos desafios existentes no estudo de QoS é o mapeamento dos parâmetros de QoS em nível de usuário (camada de aplicação) para os parâmetros baseados na tecnologia (sistema de comunicação).

Além dos requisitos especificados acima, a inclusão de estações móveis sem fio nas redes introduz alguns problemas particulares relacionados a QoS. Os enlaces sem fio apresentam pouca banda disponível em comparação às redes cabeadas. A dimensão reduzida dos dispositivos sem fio restringe a capacidade das baterias. Para economizar energia, as estações devem permanecer, sempre que possível, em um estado de baixo consumo. Além disso, suas transmissões têm alcance limitado. A livre movimentação das estações dificulta a manutenção, ao longo do tempo, dos recursos reservados para as conexões, pois as rotas podem mudar com muita freqüência. Finalmente, em redes sem fio é mais difícil garantir a segurança das comunicações, já que o ar permite o monitoramento das transmissões.

O gerenciamento da QoS [25] é a supervisão e o controle necessários para garantir que as propriedades de QoS desejadas sejam atingidas e mantidas, tanto para interações de mídia contínuas, quanto para interações discretas. Basicamente, o gerenciamento da QoS pode ser dividido em funções estáticas e dinâmicas [24]. As funções estáticas compreendem a definição dos requisitos de QoS, a negociação de um contrato entre as partes, a reserva de recursos e o controle de admissão do tráfego. As funções dinâmicas monitoram as variações ocorridas no ambiente das comunicações ao longo do tempo, possibilitando a alteração do contrato durante sua execução. O monitoramento da

QoS, o policiamento e a renegociação do contrato e a adaptação a mudanças compõem as funções dinâmicas.

O fornecimento de garantias rígidas de QoS aos tráfegos é muito difícil, sendo implementado, normalmente, através do superprovisionamento dos recursos de rede. Na maioria dos casos, as funções de gerenciamento de QoS fornecem garantias estatísticas, atuando em valores probabilísticos. Conforme citado anteriormente, a mobilidade dificulta a manutenção dos parâmetros de QoS. Enquanto para as redes fixas, podem-se fornecer garantias de QoS com probabilidades altas, para as redes móveis, isto não é possível. A solução para este problema é a adoção de um gerenciamento de QoS adaptativo, onde são especificados vários níveis de QoS aceitáveis para uma determinada aplicação. Neste caso, o gerenciamento de QoS envolve ações no nível do sistema de comunicação e o fornecimento de informações para as aplicações adaptativas.

Com relação às redes fixas, já existem várias propostas de arquiteturas bem definidas para garantir requisitos de qualidade ao tráfego como, por exemplo, a de Serviços Diferenciados [26] e a de Serviços Integrados [27], ambas desenvolvidas para serem empregadas na Internet. Já as principais questões de QoS relacionadas às redes móveis sem fio, mais especificamente às redes *ad hoc*, são bem conhecidas, mas ainda estão em aberto ou não têm solução [28, 29, 24]. Por exemplo, existe um limite superior de mobilidade e de dimensão da rede *ad hoc*, dentro do qual se podem garantir certos parâmetros de QoS.

O enfoque desta dissertação se encontra em uma das tecnologias usadas para a formação de redes *ad hoc*, o Bluetooth. As principais propostas para fornecimento e gerência de QoS nessa tecnologia serão abordadas na seção seguinte e no próximo capítulo.

2.4 Tecnologia Bluetooth

Bluetooth [4, 5] é um padrão de interface de rádio de curto alcance, criado para permitir a comunicação entre dispositivos eletrônicos sem a utilização de cabos de conexão. Posteriormente, passou a ser empregado, também, como ponto de acesso a serviços de dados e voz e na formação de redes sem fio *ad hoc*. Idealizada para ser incorporada a

pequenos equipamentos como *notebooks*, PDAs, fones de ouvido, telefones celulares e sensores, a tecnologia Bluetooth precisa satisfazer os seguintes requisitos: baixo custo, baixo consumo de energia e interface de tamanho reduzido. Em 1998, um grupo de empresas dos ramos de telecomunicações e informática criou o Bluetooth SIG (*Special Interest Group*) [4], com o objetivo de padronizar o desenvolvimento do Bluetooth. O SIG lançou uma especificação industrial aberta, atualmente na versão 1.1, contendo duas partes: o núcleo e os perfis. O núcleo define as características do rádio e a pilha de protocolos para a comunicação entre dois dispositivos. Os perfis especificam quais protocolos da pilha de protocolos do Bluetooth devem ser implementados para cada aplicação.

A pilha de protocolos Bluetooth é mostrada na Figura 2.1. Da Subseção 2.4.1 até a 2.4.6, descreve-se a pilha de protocolos, com ênfase nas camadas física e de enlace de dados. A Subseção 2.4.7 apresenta as *scatternets* e a Subseção 2.4.8 destaca os principais trabalhos relacionados às picorredes.

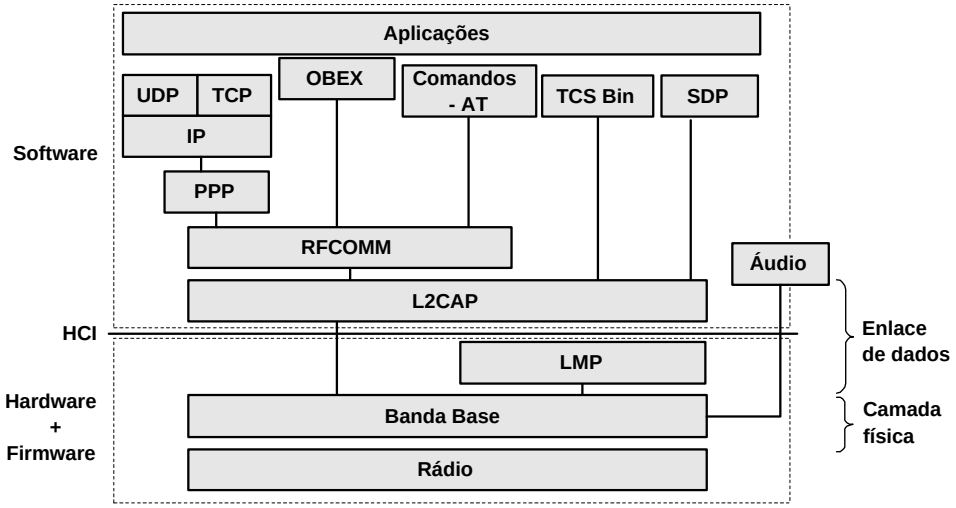


Figura 2.1: Pilha de protocolos Bluetooth.

2.4.1 Rádio Bluetooth

O rádio Bluetooth opera na banda ISM (*Industrial, Scientific and Medical*) de 2.4 GHz, disponível mundialmente sem a necessidade de licença, e emprega a técnica de saltos de freqüência com espalhamento espectral (FHSS) na transmissão. O espectro de freqüências é dividido em 79 canais de rádio-freqüência (23 em alguns países), cada

um com 1 MHz de largura de banda, e a freqüência de saltos é de 1600 por segundo. Este esquema de saltos de freqüência torna o enlace rádio mais imune a interferências externas.

Foram definidas três classes de dispositivos, segundo a potência de transmissão. A Tabela 2.1 destaca essas classes.

Classe	Alcance (m)	Potência Máxima
1	100	20dBm (100mW)
2	10	4dBm (2,5mW)
3	1	0dBm (1mW)

Tabela 2.1: Classes de transmissores Bluetooth.

2.4.2 Banda Base

A banda base é a camada física da pilha de protocolos. Situa-se acima da camada de rádio, desempenhando o papel de controladora desse enlace. É responsável, entre outras funções, pela criação das picorredes e dos enlaces. A picorrede é uma rede, com até oito integrantes ativos, formada por dispositivos que compartilham o mesmo canal, ou seja, o mesmo esquema de saltos de freqüência. Um dos dispositivos assume o papel de mestre e os outros se comportam como escravos, conforme mostra a Figura 2.2. O mestre de uma picorrede é quem dita a seqüência de saltos e a fase nesta seqüência, através do seu endereço de 48 bits e do seu relógio, respectivamente.

O tempo é dividido em intervalos de 625 microssegundos de duração chamados *slots*. A cada novo *slot*, os dispositivos mudam para a próxima freqüência da seqüência de saltos.

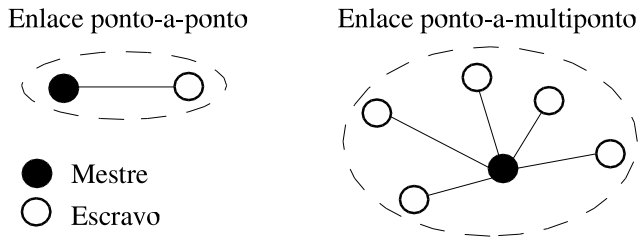


Figura 2.2: Exemplos de picorredes.

A comunicação bidirecional ocorre através de duplexação por divisão no tempo

(TDD - *time division duplex*), onde os dispositivos transmitem e recebem alternadamente. Um escravo somente poderá transmitir em um *slot* se tiver sido endereçado pelo mestre no *slot* anterior. A cada novo *slot*, os dispositivos de uma mesma picorrede mudam para a próxima freqüência da seqüência de saltos. Entretanto, caso o mestre e um escravo estejam envolvidos em uma transmissão de pacote com comprimento maior que um *slot* (os tipos de pacote são abordados adiante nesta seção), ambos permanecem na mesma freqüência até o término da transmissão. Esse procedimento está ilustrado na Figura 2.3.

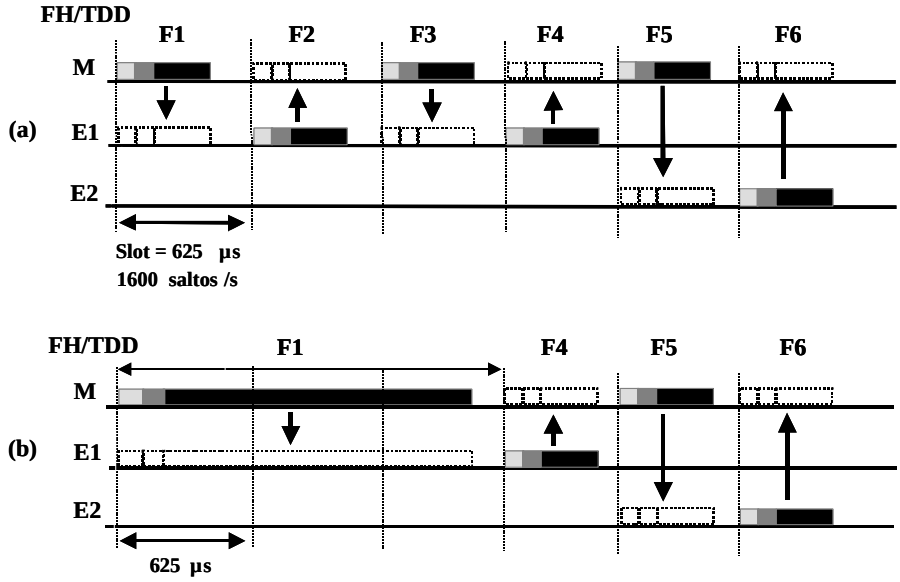


Figura 2.3: Funcionamento do FH/TDD. Todos os pacotes em (a) ocupam um *slot*. Já em (b), o mestre M transmite um pacote de três *slots* para o escravo E1. M e E1 permanecem na mesma freqüência F1 até o término da transmissão, quando, então, pulam para F4.

Há dois tipos de enlaces físicos possíveis em uma picorrede: síncrono, orientado a conexão (SCO - *Synchronous Connection Oriented*) e assíncrono, sem conexão (ACL - *Asynchronous Connectionless*). O SCO é um enlace ponto-a-ponto, simétrico, entre o mestre e um escravo, mantido pelo mestre em *slots* reservados a intervalos regulares. Sua aplicabilidade principal é o tráfego de voz. O ACL é um enlace ponto-a-multiponto, entre o mestre e todos os escravos da picorrede. Não pode ser estabelecido nos *slots* reservados para o enlace SCO. É usado, tipicamente, para transmitir dados no esquema de melhor esforço (*best effort*), e o tráfego é escalonado pelo mestre.

Existem 15 tipos de pacote na camada banda base, definidos para os enlaces SCO,

ACL ou ambos. Os pacotes ID, NULL, POLL, FHS e DM1 são usados tanto em enlaces SCO, quanto em ACL. DH1, AUX1, DM3, DH3, DM5 e DH5 são usados apenas por enlaces ACL e HV1, HV2, HV3 e DV apenas por SCO. Os comprimentos dos pacotes são de 1, 3 ou 5 *slots*. A Tabela 2.2 relaciona as principais características de cada pacote.

Os pacotes ID e FHS são usados nos procedimentos de descoberta e conexão de dispositivos. Se o mestre não tem dados a transmitir e precisa escalonar um escravo, usa o POLL. O NULL é usado pelo escravo, quando este não tem dados a transmitir, mas precisa responder ao mestre. Os pacotes HV* carregam informações de voz em enlaces SCO. Todos ocupam um *slot*, mas diferem no uso ou não de FEC (código de correção de erros, conforme será visto adiante). O tipo DV combina informações de voz e dados em um único pacote. Os DM* e DH* carregam dados em enlaces ACL. Os primeiros têm parte de sua área de dados ocupada por FEC e, portanto, carregam menos informações do que os DH*. O símbolo '*' em DM* e DH* representa a duração, em *slots*, de cada pacote.

Tipo de pacote	Tamanho (*slots*)	Espaço p/ dados (bytes)	Cabeçalho dos dados (bytes)	FEC	Taxa máx. Simétrico (kbps)	Taxa máx. Assim.	
						Ida (kbps)	Volta (kbps)
NULL	1	-	-	-	-	-	-
POLL	1	-	-	-	-	-	-
FHS	1	18	-	2/3	-	-	-
ID	1	-	-	-	-	-	-
HV1	1	10	-	1/3	64	-	-
HV2	1	20	-	2/3	64	-	-
HV3	1	30	-	-	64	-	-
DV	1	10+(0-9)D	1D	2/3D	64+57D	-	-
AUX1	1	0-29	1	-	185,6	185,6	
DM1	1	0-17	1	2/3	108,8	108,8	108,8
DH1	1	0-27	1	-	172,8	172,8	172,8
DM3	3	0-121	2	2/3	258,1	387,2	54,4
DH3	3	0-183	2	-	390,4	585,6	86,4
DM5	5	0-224	2	2/3	287,6	477,8	36,3
DH5	5	0-339	2	-	433,9	723,2	57,6

Tabela 2.2: Tipos de pacotes de banda base. Os campos marcados com 'D' no tipo DV relacionam-se apenas aos dados (e não à voz).

Todos os pacotes têm um formato genérico, mostrado na Figura 2.4. O código de acesso é a primeira parte do pacote recebida em uma transmissão. Ele é usado

na sincronização e no procedimento de conexão, entre outras funções. Existem três tipos de código: *Channel Access Code* (CAC), *Device Access Code* (DAC) e *Inquiry Access Code* (IAC). O primeiro identifica unicamente a picorrede. O DAC e o IAC são usados, respectivamente, nos procedimentos de *inquiry* e *paging*, descritos adiante nesta seção. O cabeçalho contém informações como a numeração do pacote, controle de fluxo, endereço do escravo e código de correção de erro. A área de dados vindos das camadas superiores pode conter voz, dados ou ambos. No caso de dados, o campo conterá também um cabeçalho de dados.

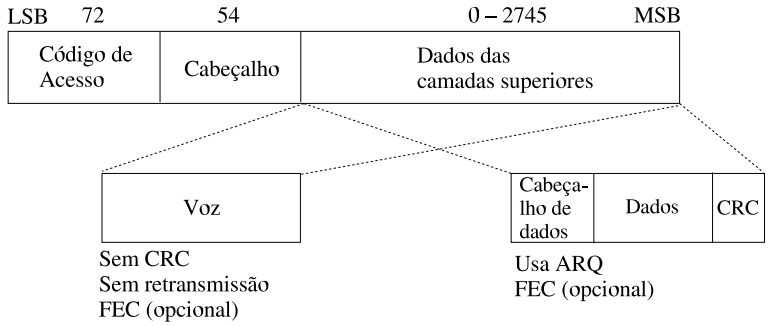


Figura 2.4: Formato do pacote Bluetooth. Tamanhos: código de acesso: normalmente 72 bits; cabeçalho: 54 bits; área de dados: de 0 a 2745 bits.

Além do esquema de saltos de freqüência, a banda base dispõe de técnicas de retransmissão de pacotes (ARQ - *Automatic Repeat Request*) e de correção de erros (FEC - *Foward Error Correction*), para fornecer robustez e confiabilidade ao Bluetooth. No caso do ARQ, os pacotes de banda base são retransmitidos até que o transmissor receba uma confirmação do recebimento (ACK). Bluetooth emprega o reconhecimento não numerado, positivo ou negativo, configurado em um bit do cabeçalho. Se o limite de retransmissões for excedido, o pacote é descartado. Existem duas codificações de FEC: com taxa a 1/3, onde cada bit é repetido três vezes, e com taxa a 2/3, onde cada grupo de 10 bits é codificado para 15. O FEC com taxa a 1/3 está presente nos cabeçalhos de todos os pacotes e é opcional na área de dados dos pacotes de voz. Já o FEC com taxa a 2/3 pode ser usado para proteger a área de dados de pacotes de voz e dados.

O controlador de enlace Bluetooth opera em dois estados principais: Em Espera (*Standby*) e Conectado (*Connected*). O estado Em Espera é o padrão de baixo consumo de energia de uma unidade Bluetooth. No estado Conectado, mestre e escravo podem trocar pacotes. Na transição do estado Em Espera para o Conectado, existem alguns

estados intermediários, usados para formar uma picorrede e para adicionar novos escravos a uma picorrede já existente. A Figura 2.5 apresenta o diagrama de estados do controlador de enlace.

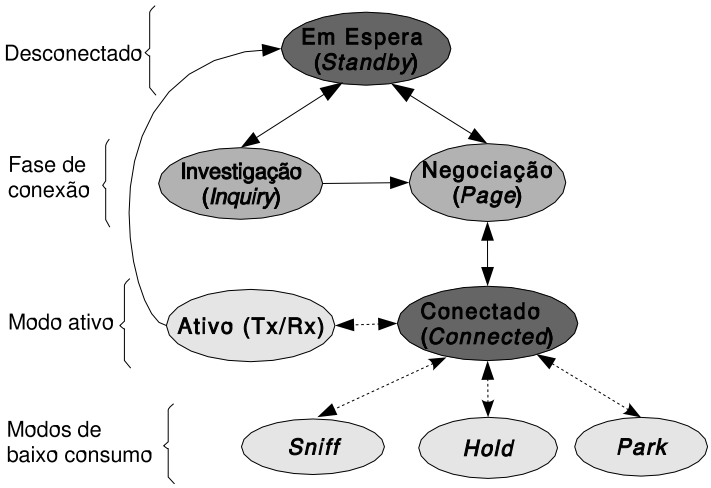


Figura 2.5: Diagrama de estados do controlador de enlace.

Para descobrir os dispositivos em sua proximidade, uma estação executa o procedimento de investigação (*inquiry*). Nessa fase, os dispositivos trocam as informações de seus endereços únicos de 48 bits (BD_ADDR) e de seus relógios. O dispositivo que iniciar o procedimento de investigação tornar-se-á o mestre da picorrede. As unidades que responderem à solicitação serão os escravos. Terminada esta etapa, os dispositivos voltam para o estado Em Espera, agora com o conhecimento de seus pares, e segue-se uma fase de negociação da conexão (*paging*), novamente iniciado pelo futuro mestre.

A diferença funcional entre os procedimentos de investigação e de negociação da conexão está no uso de uma seqüência de saltos de freqüência universal (comum a todos os dispositivos) no primeiro e uma seqüência estabelecida por cada par de dispositivos, ponto-a-ponto, no segundo. No procedimento de negociação da conexão, o futuro mestre e cada escravo negociam o canal a ser usado na comunicação. Finalmente, os dispositivos mudam para o canal negociado e passam para o estado Conectado. Uma vez conectado à piconet, o escravo recebe e passa a usar um endereço de 3 bits chamado *Active Member Address* (AM_ADDR). Vale ressaltar que, em qualquer instante do procedimento de conexão, um dispositivo pode voltar para o estado Em Espera caso o processo de conexão seja interrompido.

Como uma das preocupações do Bluetooth é a economia de energia, existem quatro modos de operação para os dispositivos no estado Connectado: Ativo, *Sniff*, *Hold* e *Park*. No modo Ativo, a comunicação pode ocorrer normalmente. Os outros três caracterizam modos de baixo consumo de energia. No modo *Sniff*, o escravo dorme durante um intervalo pré-definido e acorda, periodicamente, para escutar as transmissões do mestre. É o modo de menor economia dos três definidos para baixo consumo. No modo *Hold*, o escravo entra em baixo consumo por um intervalo de tempo fixo, após o qual volta à atividade normal. Não há uma periodicidade como no modo *Sniff*. Cada vez que um escravo pretende entrar em *Hold*, precisa negociar a duração com o mestre. Finalmente, no modo *Park*, o escravo libera seu endereço de dispositivo ativo (AM_ADDR), mas permanece sincronizado à piconet, utilizando um endereço de 8 bits chamado *Parked Member Address* (PM_ADDR). Um dispositivo em modo *park* não pode participar da comunicação de dados na picorrede. Como uma picorrede pode ter no máximo sete escravos, o modo *park* permite que mais escravos continuem sintonizados à picorrede, para uma possível reativação futura.

2.4.3 Protocolo de Gerência do Enlace (LMP)

O LMP (*Link Manager Protocol*) encontra-se na camada de enlace de dados e é responsável pela configuração e gerenciamento das conexões banda base. Por exemplo, os procedimentos de autenticação e criptografia e os modos de operação de baixo consumo de energia são definidos na banda base, mas são configurados e ativados através de transações LMP entre dois dispositivos.

O LMP consiste de um conjunto de mensagens de controle que são trocadas entre dispositivos, baseadas no endereço AM_ADDR. Essas mensagens são sempre enviadas em pacotes de um *slot* e têm prioridade sobre os pacotes de dados vindos das camadas superiores. As mensagens podem ser agrupadas em três funções: gerenciamento de picorrede, configuração do enlace e segurança.

2.4.4 Interface de Controle do *Host* (HCI)

A pilha de protocolos Bluetooth foi dividida de forma que uma parte fosse implementada em software (*host*) e outra em hardware (dispositivo físico e *firmware*). Isso não impede a produção de dispositivos com a tecnologia desenvolvida totalmente em hardware. A HCI (*Host Controller Interface*) é constituída de três partes: um módulo de *driver* no *host*, um de *firmware* e um canal de transporte entre os dois anteriores. Sua função é fornecer ao *host* uma interface de comandos para se fazer acesso ao gerenciador de enlace e ao controlador de banda base e para se conhecer o estado do hardware e dos registradores de controle. De um modo geral, essa interface provê um método uniforme de acesso às funcionalidades da banda base.

2.4.5 Protocolo de Adaptação e Controle do Enlace Lógico (L2CAP)

L2CAP (*Logical Link Control and Adaptation Protocol*) é o protocolo da camada de enlace de dados que fornece serviços para as camadas superiores e esconde dessas os detalhes de implementação das camadas inferiores. O protocolo foi definido apenas para enlaces ACL e trabalha com o conceito de canais lógicos. Cada extremidade de um canal lógico é referenciado por um identificador local (CID - *channel identification*). Os canais podem ser sem conexão ou orientados a conexão. Nos orientados a conexão, para cada fluxo L2CAP é estabelecido um canal entre as duas extremidades. A numeração de uma extremidade realizada por um dispositivo é independente daquela utilizada por seu par, excetuando-se os canais reservados para sinalização. Os canais sem conexão restringem o fluxo de dados a um único sentido. Neste caso, o CID na origem representa um grupo de dispositivos remotos. Pode haver vários canais lógicos entre duas estações, mas todos sobre um mesmo enlace ACL.

Como o protocolo de banda base não tem como identificar os protocolos de camada superior, L2CAP realiza a multiplexação desses no enlace. Também realiza a segmentação e remontagem de pacotes (SAR), função necessária para permitir a utilização, por parte das camadas superiores, de pacotes maiores que o tamanho máximo aceito pela banda base.

2.4.6 Camadas Superiores

Os principais protocolos que se situam sobre o L2CAP são: protocolo de descoberta de serviços (SDP - *service discovery protocol*), emulação de porta serial (RFCOMM) e especificação de controle de telefonia binário (TCS *binary* - *telephony control protocol specification* - *binary*). SDP fornece uma maneira das aplicações descobrirem os serviços disponibilizados na proximidade. O ambiente Bluetooth requer um SDP específico, já que o conjunto de serviços muda dinamicamente com a movimentação dos dispositivos. RFCOMM realiza a emulação de portas seriais RS-232 (EIA/TIA 232-E), para aplicações que necessitam desta funcionalidade. Finalmente, TCS-*binary* define a sinalização de controle de chamadas para o estabelecimento de chamadas de voz e dados entre dispositivos Bluetooth.

2.4.7 As *Scatternets*

Várias picorredes podem coexistir em uma mesma área de cobertura, pois seus padrões de saltos de freqüência são mutuamente ortogonais. Isto permite que se obtenha uma vazão agregada bem superior à de uma picorrede individual. Entretanto, se dispositivos de picorredes distintas quiserem se comunicar, pode haver a interconexão destas redes, criando uma topologia conhecida por *scatternet*. A *scatternet* se forma quando ao menos uma estação participa de duas ou mais picorredes. Essa participação se baseia em uma divisão no tempo, ou seja, em qualquer instante a estação (aqui chamada de estação ponte) pode estar ativa em somente uma picorrede. A estação ponte pode desempenhar o papel de escravo em várias picorredes, mas somente pode ser mestre em uma, pois o endereço do mestre serve como base para o cálculo do esquema de saltos de freqüência e esse esquema não pode ser repetido. A Figura 2.6 apresenta a estrutura de uma *scatternet*.

A ponte pode trabalhar com aplicações independentes em cada picorrede ou pode encaminhar pacotes de uma picorrede para outra. O primeiro caso está exemplificado na Figura 2.7 (a), onde uma impressora é compartilhada por duas picorredes distintas. A Figura 2.7 (b) apresenta dois exemplos do segundo caso, sendo um a transmissão de vídeo entre picorredes e o outro a comunicação de voz via *scatternet*, alcançando uma rede infra-estruturada através do ponto de acesso.

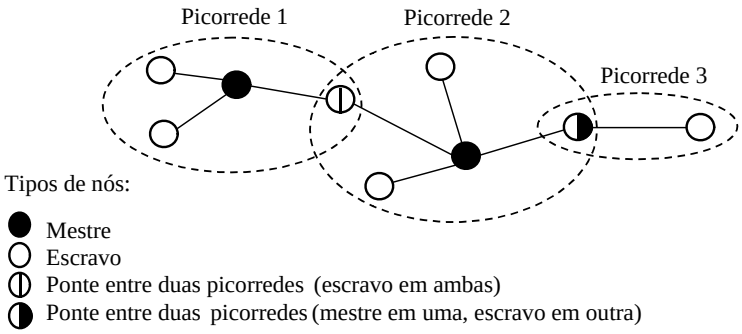


Figura 2.6: Exemplo da estrutura de uma *scatternet*.

Ao mesmo tempo em que propiciam versatilidade ao Bluetooth, as *scatternets* trazem consigo novas questões relativas à sua implementação. Essas questões são exploradas no Capítulo 3, com ênfase na parte de escalonamento, um dos objetivos principais desta dissertação.

2.4.8 Pesquisas Recentes em Bluetooth

A pesquisa em Bluetooth pode ser dividida em dois grupos principais, segundo a topologia da rede: estudos em picorredes e em *scatternets*. O segundo grupo será explorado ao longo do Capítulo 3.

A maioria das pesquisas para melhorar o aproveitamento do enlace Bluetooth em picorredes se concentra em propor variantes do mecanismo de escalonamento *round robin*. O estudo de mecanismos de escalonamento não é recente. Em 1992, Liu *et al.* [7] publicaram um estudo sobre mecanismos de escalonamento e mostraram que as técnicas de escalonamento exaustivas são ótimas em sistemas simétricos. Estes resultados não podem ser empregados diretamente às picorredes, pois elas apresentam algumas peculiaridades. Por exemplo, o esquema de acesso ACL (sem conexão) é controlado pelo mestre, onde ele escalona os seus escravos, realizando o *polling* dos mesmos. Após a transmissão do mestre para o escravo, o escravo tem sempre a chance de transmitir pacotes no sentido reverso. Além disso, o mestre conhece as suas filas, mas não as dos escravos.

Em uma situação ideal, supõe-se que o mestre conheça as filas de todos os seus escravos, o que evita o desperdício de *slots* [11]. Este tipo de estudo é válido como parâmetro de comparação. O trabalho de Capone *et al.* [9] mostra um estudo

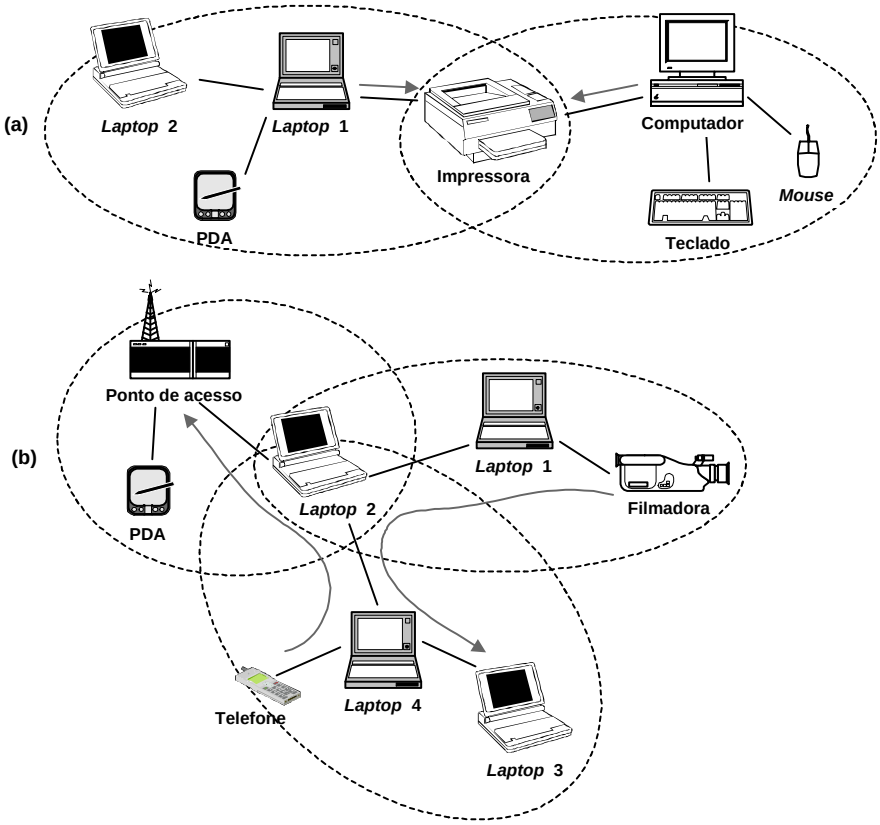


Figura 2.7: Exemplos de cenários de *scatternets*. Em (a), a ponte (impressora) recebe pacotes de ambas as picorredes. Em (b), a ponte (*laptop* 2) encaminha pacotes entre as picorredes.

comparativo entre esquemas de *polling* e propõe uma variante para o *Weighted Round Robin* (WRR), o *Limited Weighted Round Robin* (LWRR). Os autores concluem que, em termos de vazão, o LWRR chega próximo ao resultado ideal, não sendo, portanto, interessante aumentar a complexidade dos escalonadores na tentativa de obter novos ganhos. Das *et al.* [10] introduzem o *Adaptive Flow-based Polling* (AFP). Ambas as propostas variam a ordem de serviço aos escravos, conforme a disponibilidade de dados em suas filas, minimizando a passagem por escravos que não tenham dados a transmitir. Em contrapartida, necessitam da criação de parâmetros adicionais a serem transmitidos com os dados. Johansson *et al.* [30] comparam os algoritmos *round robin*, *exhaustive polling* e *fair exhaustive polling*, concluindo que o desempenho aumenta com a utilização de pacotes de múltiplos *slots*. Entretanto, o estudo adota simplificações, tais como probabilidades de erro constantes e *buffers* ilimitados.

De um modo geral, as propostas de escalonamento intrapicorrede citadas buscam, através de mecanismos adaptativos, otimizar a ocupação do enlace ACL, evitando o

desperdício de banda. A especificação Bluetooth prevê o uso de enlace SCO, separadamente, para o tráfego com requisito de retardo limitado, como o tráfego de voz. Entretanto, as garantias fornecidas para o enlace SCO são obtidas ao custo de uma grande perda na vazão agregada do enlace ACL [10]. Ainda assim, os autores das propostas existentes não se preocupam muito em estudar a possibilidade de se compartilhar o enlace ACL entre o tráfego de melhor esforço e o de requisito de retardo limitado.

Os mecanismos de segmentação e remontagem (SAR), retransmissão (ARQ) e correção de erros (FEC) também são alvos de estudos. SAR influi no percentual de ocupação do meio físico. Das *et al.* [10] propõem um esquema de SAR que tenta minimizar o desperdício de banda, escolhendo pacotes de banda base convenientes (*best fit*) para segmentar os pacotes L2CAP. No estudo de Kalia *et al.* [11], abre-se a possibilidade de transmitir o segundo pacote de uma fila antes do primeiro, caso este não se encaixe nos *slots* disponíveis. Para que o receptor saiba o pacote que está recebendo, é necessário o envio da informação adicional. Os mecanismos de ARQ e FEC estão relacionados com problemas de interferência e evitam que seus efeitos cheguem à camada de transporte. A interferência pode ocorrer tanto através da existência de múltiplas picorredes na mesma área [31], quanto pela coexistência de tecnologias que operam na mesma faixa de freqüência como Bluetooth e IEEE 802.11 [32].

2.5 Resumo

Neste capítulo foram apresentados os conceitos básicos de redes móveis sem fio e de qualidade de serviço nestas redes. As redes móveis sem fio foram classificadas em redes infra-estruturadas e redes *ad hoc*. Uma das tecnologias propostas para a criação dessas últimas é o Bluetooth, assunto que foi abordado em seguida. Bluetooth constitui a tecnologia-base sobre a qual foi desenvolvida esta dissertação e, portanto, foi explorada com maior nível de detalhamento. As camadas física e de enlace da pilha de protocolos do Bluetooth foram descritas, bem como sua topologia básica de rede, a picorrede. As *scatternets*, criadas através da interconexão de picorredes, também foram citadas. Entretanto, devido à quantidade de questões ainda em aberto sobre estas redes e à relevância deste assunto para a dissertação, o próximo capítulo dedica-se ao estudo das *scatternets*.

Capítulo 3

Scatternets: Questões e Propostas Existentes

3.1 Introdução

A versão 1.1 da especificação Bluetooth conceitua as *scatternets* [33], porém, não padroniza os algoritmos e mecanismos necessários para a comunicação entre dispositivos pertencentes a picorredes distintas. Por isso, a maior parte dos produtos disponíveis atualmente no mercado não permite a formação de *scatternets*. Para resolver o problema, foi criado o perfil PAN (*Personal Area Network*) [34], para permitir que dois ou mais dispositivos formem uma rede *ad hoc*, e seu desenvolvimento foi dividido em duas fases. A primeira, já divulgada, aborda as picorredes de forma individual e a segunda, ainda sem previsão de conclusão, trata das *scatternets*.

A inexistência de padronização relacionada às *scatternets* tem possibilitado várias pesquisas na área. De um modo geral, essas pesquisas se encaixam nos seguintes tópicos: formação da topologia, roteamento ou encaminhamento de pacotes e escalonamento interpicorrede. A Seção 3.2 aborda o tópico de formação da topologia. A seção seguinte trata dos mecanismos de encaminhamento de pacotes. A Seção 3.4 explora o escalonamento da estação ponte entre suas picorredes, também chamado de escalonamento interpicorrede. Este assunto foi abordado em detalhes, por estar diretamente relacionado aos objetivos desta dissertação. Finalmente, a Seção 3.5 conclui o capítulo.

3.2 Formação da *Scatternet*

A topologia da *scatternet* influi diretamente em parâmetros da rede como vazão, retardo fim-a-fim e consumo de energia. A formação ótima significa aquela que obtém o melhor desempenho para uma determinada métrica. Portanto, a topologia ótima para vazão pode ser diferente daquela que minimiza o retardo. Além desses fatores, os algoritmos de formação de *scatternets* devem se preocupar com o tempo gasto para alcançar uma topologia inicial estável.

A política de formação da *scatternet* deve se preocupar com a topologia ao longo do tempo. A mobilidade nas redes *ad hoc* permite que estações entrem e saiam da rede aleatoriamente. A distribuição do tráfego pode mudar dinamicamente. Essas variações, muitas vezes, obrigam o algoritmo de formação a atualizar a topologia da rede, com o objetivo de manter suas conexões e de evitar a degradação das métricas especificadas. A impossibilidade de comunicação direta entre escravos também pode dificultar a difusão das informações sobre os enlaces. Além disso, o algoritmo de roteamento utilizado, se reativo ou pró-ativo, deve ser considerado na formação da rede ao longo do tempo. Por exemplo, a procura por um novo caminho em um algoritmo de roteamento reativo pode ser usada pelo algoritmo de formação para adaptar a topologia da *scatternet*. Essas considerações mostram que não há uma solução trivial para o problema da formação de topologias eficientes.

A topologia de uma picorrede depende, basicamente, da escolha de um mestre. Uma vez definido o mestre, os dispositivos próximos tornam-se escravos, gerando uma topologia em estrela. Já a formação de uma *scatternet* envolve mais fatores, aumentando sua complexidade. Por exemplo, basta que um nó comute entre duas picorredes para que uma *scatternet* esteja caracterizada. Um único nó pode estar presente em várias picorredes, contanto que seja mestre em, no máximo, uma. Da mesma forma, uma picorrede pode ter vários de seus nós compartilhados com outras picorredes. Portanto, fixada a quantidade de nós, o espaço de topologias possíveis para *scatternets* é muito grande. Isso está caracterizado no trabalho de Bhagwat et al. [35]. Uma das primeiras tentativas de se obter uma topologia ótima foi apresentada por Miklos et al. [36]. Devido à complexidade do problema, os autores adotam uma abordagem estatística, na qual são geradas topologias aleatórias e são estudados os efeitos delas no desempenho

do sistema.

Vários trabalhos abordam a formação das *scatternets*. O algoritmo BTCP (*Bluetooth Topology Construction Protocol*) [37] é composto de duas fases: eleição de um líder (nó que terá o conhecimento de todas as outras estações) e distribuição de funções às estações pelo líder. Esse algoritmo funciona bem quando todas as estações são ligadas, aproximadamente, no mesmo tempo. Abordagem similar à anterior é apresentada no algoritmo de Ramachandran et al. [38]. A rede é particionada em várias picorredes, e um nó é eleito "supermestre". Entretanto, as picorredes não chegam a ser interconectadas. Law et al. [39] criam um algoritmo de formação composto por uma fase única. Todos os dispositivos são iniciados como líderes. Conforme vão se agrupando, alguns nós deixam de ser líderes, permanecendo apenas um como líder em cada grupo. Quando restar apenas um líder, a *scatternet* estará formada. O uso de duas fases para formar a *scatternet* [37] torna o algoritmo um pouco mais lento que a proposta com uma única fase [39]; entretanto, permite maior flexibilidade na construção da topologia. Finalmente, existem algumas propostas que constroem redes como estruturas de árvores [40, 41]. O comportamento hierarquizado em pais e filhos traz simplificações ao algoritmo de roteamento da rede, ao custo de algumas restrições como a não-existência de ciclos na topologia ou o limite da quantidade de picorredes a que uma estação pode pertencer.

3.3 Encaminhamento de Pacotes entre Picorredes

O suporte ao TCP/IP é necessário para que redes Bluetooth possam comunicar-se com a Internet e diversas aplicações já existentes sejam reaproveitadas. A especificação Bluetooth prevê a utilização do TCP/IP sobre PPP para que os dispositivos Bluetooth estabeleçam conexões ponto-a-ponto com um ponto de acesso à Internet. Entretanto, com o emprego da tecnologia Bluetooth em redes *ad hoc*, muitos pesquisadores têm estudado outras maneiras de se configurar o protocolo IP na pilha de protocolos.

A posição do protocolo IP na pilha Bluetooth influi na maneira como se pretende realizar o roteamento de pacotes na rede. Existem duas correntes básicas quanto ao roteamento de pacotes no Bluetooth: encaminhamento em nível 2 (camada de enlace

Bluetooth) ou roteamento em nível 3 (camada de redes IP).

As propostas de encaminhamento de pacotes em nível 2, portanto, abaixo da camada de redes IP, sugerem que a camada de enlace Bluetooth seja a responsável por direcionar os pacotes ao longo dos saltos, desde a estação de origem até o destino. A motivação para esta forma de encaminhar os pacotes está no fato de as *scatternets* apresentarem características diferentes das redes tradicionais.

O roteamento em nível 3 prevê o uso de algoritmos baseados no endereçamento IP para realizar o roteamento dos pacotes em uma rede de múltiplos saltos. Esta é a forma tradicional de encaminhamento de pacotes empregada na Internet. Ambas as técnicas serão abordadas nas duas subseções seguintes.

3.3.1 Soluções Abaixo da Camada de Redes IP

O perfil PAN [34] está sendo desenvolvido pelo Bluetooth SIG e descreve como dois ou mais dispositivos Bluetooth podem formar uma rede *ad hoc* e, também, como uma rede remota pode ser endereçada através de um ponto de acesso. Esse perfil utiliza o *Bluetooth Network Encapsulation Protocol* (BNEP) [42] para o envio de pacotes. BNEP é um protocolo de redes Bluetooth, situado abaixo da camada de rede IP. A Figura 3.1 mostra a pilha de protocolos Bluetooth com BNEP.

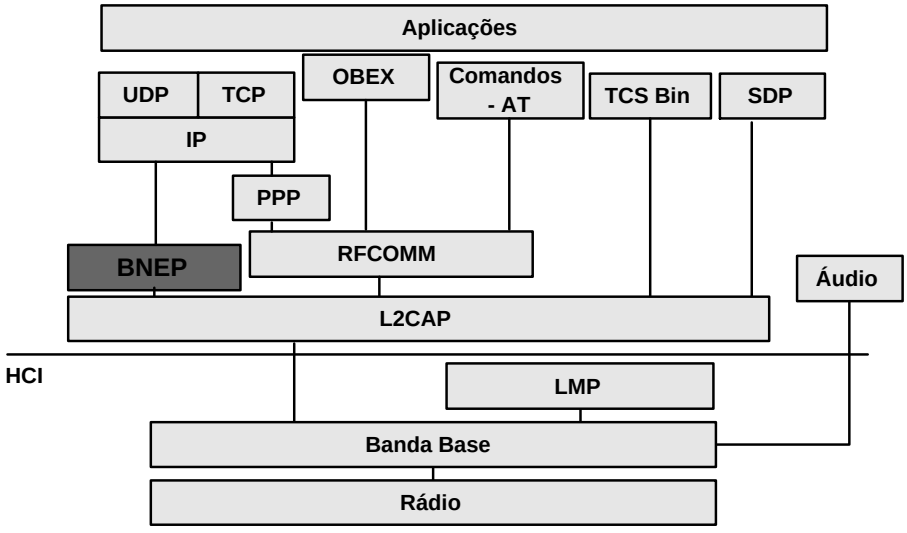


Figura 3.1: Pilha de protocolos Bluetooth, incluindo BNEP sobre o L2CAP.

BNEP funciona, para as camadas superiores, como uma interface Ethernet. Assim,

o protocolo permite a criação de um segmento de rede único envolvendo uma ou mais picorredes, conforme mostra a Figura 3.2, adaptada do trabalho de Johansson et al. [43]. Além disso, mecanismos IP bastante conhecidos como DHCP e ARP, dependentes da conectividade no nível de enlace, podem ser empregados sem modificações. BNEP utiliza o endereço MAC da interface Bluetooth para endereçar as estações que estejam em picorredes distintas da estação de origem.

Em uma primeira fase, o perfil PAN define a utilização do BNEP apenas para formar segmentos de rede dentro de uma picorrede. A segunda fase prevê sua expansão para as *scatternets*. Quando ambas as fases estiverem concluídas, será possível empregar os algoritmos de roteamento tradicionais para redes IP em redes Bluetooth, abstraindo-se da implementação das camadas física e de enlace Bluetooth.

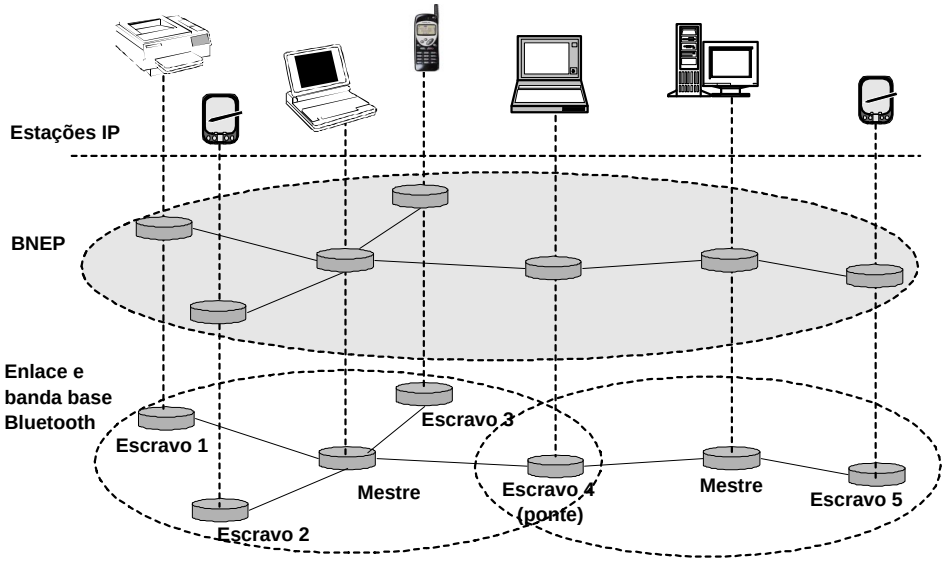


Figura 3.2: Possível utilização do protocolo BNEP na formação de um segmento de rede único para a camada IP (adaptado de [43]).

Uma proposta interessante, independente da solução proposta pelo perfil PAN, foi adotada por Raman et al. [44]. Na seção anterior, mostrou-se que a formação da *scatternet* influi diretamente no roteamento dos pacotes. Visando otimizar o desempenho dos tráfegos de uma *scatternet*, Raman et al. sugerem a integração de protocolos de vários níveis em uma camada única. Mais especificamente, os autores agrupam as funcionalidades de formação de enlaces, roteamento de pacotes e descoberta de serviços, permitindo, por exemplo, que a topologia da *scatternet* seja otimizada para um serviço específico. Como as decisões são tomadas com o conhecimento das necessidades de

cada camada, consegue-se otimizar o desempenho da rede. Em contrapartida, perde-se a independência entre os níveis, necessária à interoperabilidade entre protocolos distintos.

3.3.2 Soluções na Camada de Redes

Os pesquisadores adeptos do roteamento tradicional nas *scatternets* defendem o aproveitamento dos algoritmos de roteamento para redes *ad hoc* já definidos pelo grupo de trabalho IETF MANET [21]. A camada de rede IP estaria situada diretamente sobre a camada de enlace Bluetooth. Esta posição é compartilhada por Atwal [45]. Ele alega que a inclusão da camada intermediária BNEP entre IP e L2CAP acrescenta um novo cabeçalho aos pacotes, reduzindo a taxa de transmissão de dados dos usuários e gerando maior carga de processamento. Entretanto, também existem algumas questões em aberto relativas ao uso do IP sobre L2CAP.

Devido às peculiaridades do Bluetooth, o roteamento MANET parece ser superdimensionado para as PANs. As *scatternets* se caracterizam como redes pequenas e/ou com mobilidade limitada. Por outro lado, as propostas de roteamento MANET incluem sinalizações ou informações adicionais no cabeçalho dos pacotes, para garantir a escalabilidade e padrões de mobilidade. A proposta RVM (*Routing Vector Method*) [46] é uma alternativa aos algoritmos MANET, que procura minimizar a quantidade de informações nos roteadores, transportando a rota diretamente nos pacotes.

Outra dificuldade de se colocar a camada IP diretamente sobre o enlace L2CAP vem das características particulares das redes *ad hoc* formadas com Bluetooth. Por exemplo, no Bluetooth, o tráfego sempre passa pelo mestre. Nas camadas inferiores, a rede *ad hoc* é formada por um conjunto de redes menores (as picorredes), onde os dispositivos de redes distintas não se conhecem. Além disso, o protocolo IP necessita de funcionalidades de nível 2, normalmente providas pelo protocolo Ethernet, as quais não são disponibilizadas pelo L2CAP.

Independentemente do algoritmo de roteamento utilizado, se não houver um protocolo intermediário como BNEP, a solução IP para as *scatternets* precisará sofrer modificações para conter campos específicos de Bluetooth. Por exemplo, o método

RVM supracitado cria um campo de identificador para cada picorrede, possibilitando que elas sejam endereçadas ao longo da rota.

3.4 Escalonamento Interpicorrede

O mestre de uma picorrede realiza o *polling* de seus escravos, através de um algoritmo de escalonamento, caracterizando o escalonamento intrapicorrede. Dessa forma, o mestre tem o controle do tráfego interno à picorrede. Em uma *scatternet*, ao menos uma estação, denominada ponte, pertence a mais de uma picorrede. Essa estação pode ser um escravo em múltiplas picorredes, mas mestre em apenas uma. Como as estações possuem, normalmente, apenas uma interface Bluetooth, a ponte é obrigada a dividir, no tempo, sua presença entre as picorredes. A forma como essa divisão ocorre é chamada escalonamento interpicorrede.

O grande desafio do escalonamento interpicorrede é garantir que o tráfego atravesse a ponte da maneira mais eficiente possível, sem, contudo, degradar o tráfego interno às picorredes. No funcionamento ideal, a ponte sempre estará presente em uma picorrede no momento em que o mestre a escalonar. Além disso, o mestre deve retirar a ponte de seu processo de *polling*, quando ela estiver ausente (conectada a outra picorrede). A Figura 3.3, adaptada da apresentação de Johansson et al. [43], identifica os tipos de escalonamento em uma *scatternet*.

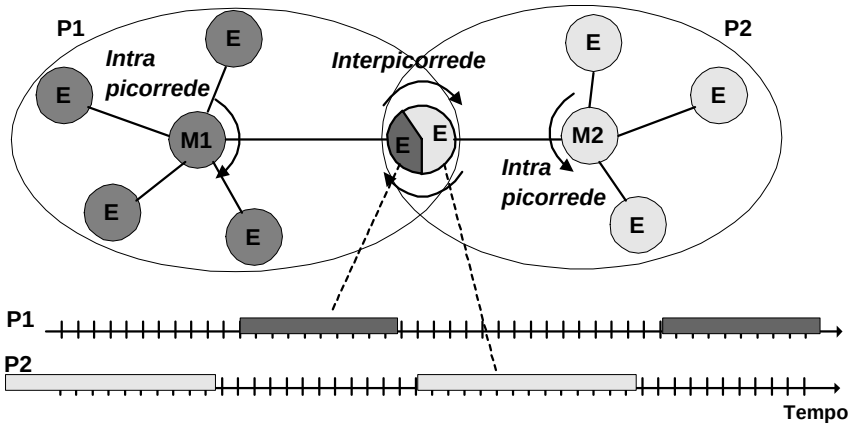


Figura 3.3: Escalonamentos intrapicorrede e interpicorrede em uma *scatternet* formada por duas picorredes (figura adaptada de [43]).

Algumas características intrínsecas do funcionamento do Bluetooth dificultam o escalonamento interpicorrede. O mestre só transmite para os escravos em *slots* pares. Portanto, ao comutar entre picorredes, a ponte precisa estar presente no próximo *slot* par, caso queira participar da comunicação na nova picorrede. Se a sincronização ocorrer durante esse *slot*, a ponte deverá esperar até o próximo *slot* par para, possivelmente, ser escalonada. Considerando que os relógios dos mestres não estejam sincronizados, até dois *slots* de tempo podem ser perdidos nessa comutação. Pode existir, também, uma pequena diferença na velocidade dos relógios dos mestres, o que causa, decorrido um certo tempo, uma "ultrapassagem" de *slots* de uma picorrede em relação a outra (em inglês, chamado *clock drift*). Por isso, o escalonamento deve, periodicamente, recalibrar seu relógio.

Uma questão importante foi resolvida empregando-se como artifício os modos de baixo consumo de energia. Se um escravo não responde ao mestre ao ser escalonado, o mestre pode desconectá-lo por abandono à picorrede. Entretanto, este escravo pode ser uma estação ponte que esteja se comunicando com outra picorrede no referido instante. Para evitar essa situação, o escravo deve entrar, após acordo com o mestre, em um dos modos de baixo consumo de energia (*Hold, Sniff* ou *Park*), durante o período de sua ausência. O escravo não estará necessariamente em modo de baixo consumo. O artifício serve apenas para que o mestre não escalone o escravo durante um período determinado.

3.4.1 Classificação dos Algoritmos de Escalonamento

Como os escravos de uma picorrede sempre escutam as transmissões de seu mestre, os algoritmos tradicionais de escalonamento como, por exemplo, o *round robin* [47] e suas variantes, o *deficit round robin* [48], o *exhaustive round robin* [7] e o *weighted round robin* [49], podem ser aplicados, com pequenas adaptações, ao escalonamento intrapicorrede.

No escalonamento interpicorrede, a estação ponte tem que alterar sua seqüência de saltos de freqüência a cada mudança de picorrede. Por isso, é desejável que a ponte permaneça escutando uma mesma picorrede por um certo período de tempo antes de mudar para outra rede. A determinação deste período tem influência direta

na vazão agregada dos tráfegos que atravessam a ponte, no retardo dos mesmos e no consumo de energia da ponte. Portanto, o escalonamento interpicorrede apresenta certas peculiaridades que o distinguem dos mecanismos tradicionais.

Os algoritmos de escalonamento interpicorrede podem ser divididos, basicamente, em duas categorias, segundo a responsabilidade pela coordenação dos mesmos:

- **mecanismos com decisão isolada** – a ponte decide unilateralmente sobre sua presença nas picorredes e pode comunicar (ou não) esta decisão aos mestres das picorredes de que faz parte como escravo;
- **mecanismos com decisão distribuída** – as decisões sobre os futuros encontros entre a ponte e as picorredes de que participa como escravo ocorrem através de acordos entre as partes.

Ambas as abordagens apresentam vantagens e desvantagens, as quais são destacadas na Tabela 3.1. A maior vantagem dos algoritmos de decisão isolada é a sua simplicidade, requerendo poucas modificações na especificação Bluetooth. Isto os torna realizáveis em curto prazo. Por outro lado, os algoritmos de decisão distribuída possuem como grande atrativo a possibilidade de se atingir uma coordenação global entre as picorredes, o que proporciona ganhos em termos de vazão agregada da *scatternet*. A idéia de coordenação global está representada na Figura 3.4. No exemplo, o mestre M2 pode entrar em acordo com as pontes para que estas participem de sua picorrede de forma alternada. A coordenação global é conseguida ao custo de sinalização adicional e maior complexidade dos algoritmos de escalonamento.

Os mecanismos de escalonamento intrapicorrede e interpicorrede também podem ser classificados como algoritmos de coordenação estática ou dinâmica. Na coordenação estática, os escalonadores dividem, através de uma determinada política, a participação das estações na comunicação e esta configuração não muda ao longo do tempo. Já na coordenação dinâmica, os escalonadores medem, periodicamente, a ocupação dos enlaces. De acordo com esta ocupação, o intervalo de tempo dedicado a cada estação é ajustado. O esquema estático é mais simples e foi empregado nos primeiros resultados sobre escalonamento. Além disso, é uma forma de fornecer garantias determinísticas para os enlaces. Entretanto, os cenários de redes *ad hoc* contemplam a mobilidade

Tipo de Coordenação	Aspectos Positivos	Aspectos Negativos
Isolada	Implementação mais simples	
	Dados específicos do escalonamento interpicorrede armazenados apenas pela ponte	Uma ponte ignora a existência das outras
	Processamento concentrado na ponte	Não há coordenação global entre as picorredes
	Sem necessidade de sinalização especial entre as estações	Tende a ser menos eficiente
	Implementado com poucas modificações na especificação	
Distribuída	Decisões de comum acordo entre mestres e pontes (quando escravas)	Processamento adicional nos mestres, além do já existente nas pontes
	Troca de mensagens entre mestres e pontes permite a coordenação global entre as picorredes	Necessidade de sinalização específica para o escalonamento
	Escalonamento mais eficiente	Mais mudanças na especificação, dificultando sua implementação

Tabela 3.1: Vantagens e desvantagens dos tipos de escalonamento interpicorrede.

das estações, diferentes categorias de tráfego e tráfegos com taxas variáveis ao longo do tempo. A adaptabilidade dos algoritmos de escalonamento é fundamental para garantir a ocupação eficiente do canal e evitar o desperdício de energia. Seguindo essa idéia, as principais pesquisas recentes sobre escalonamento levam em consideração a dinâmica dos tráfegos nas *scatternets*, desconsiderando as propostas estáticas.

3.4.2 Principais Propostas Existentes

Os trabalhos iniciais abordavam o escalonamento em *scatternets* de uma forma genérica, partindo de topologias muito simplificadas [50], preocupando-se em estudar o comportamento do tráfego e não exatamente propondo algoritmos de escalonamento.

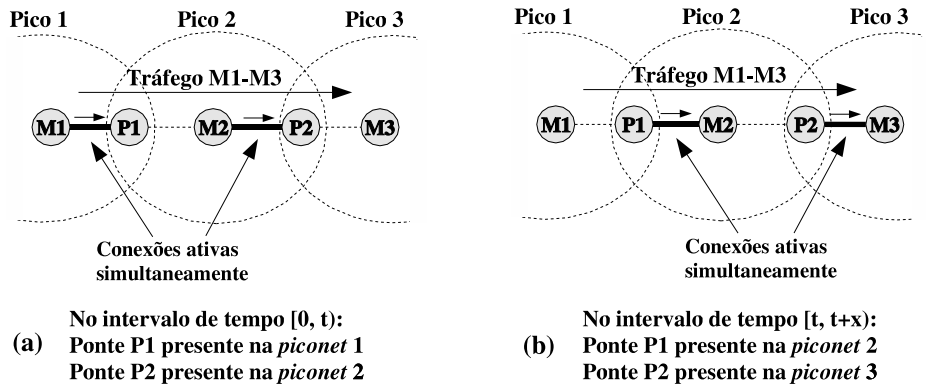


Figura 3.4: Com as pontes P1 e P2 alternando sua presença na picorrede 2, o mestre M2 pode se dedicar integralmente a uma ponte de cada vez, maximizando a vazão. Caso P1 e P2 estivessem presentes simultaneamente na picorrede 2, uma das pontes ficaria sempre ociosa.

O artigo de Johansson et al. [43] apresenta o termo "ponto de encontro" (*rendez-vous point*) como sendo o *slot* em que a ponte e o mestre de uma picorrede decidem se comunicar. Uma vez acordado o ponto de encontro, o mestre tentará escalonar a ponte nesse *slot*. Já a ponte estará sincronizada com a picorrede deste mestre no *slot* determinado. O artigo também define a "janela de encontro" (*rendez-vous window*), que é, basicamente, a duração do encontro. A partir do nível de comprometimento dos dispositivos a esses dois conceitos, os autores propõem algumas categorias de algoritmos. O objetivo do artigo é fornecer uma visão geral das questões relacionadas às *scatternets*, incluindo o tópico de escalonamento.

Seguindo a classificação apresentada na Seção 3.4.1, as propostas contidas em [12, 13, 41, 51] apresentam mecanismos de decisão distribuída, ou seja, há alguma forma de sinalização e de acordo entre a ponte e seus vizinhos, de modo que as decisões sobre o escalonamento não são tomadas exclusivamente pela ponte. Já as propostas de [14, 15, 16] apresentam algoritmos de decisão isolada. De maneira geral, as propostas nas duas categorias procuram criar mecanismos adaptativos, onde as variações de tráfego ao longo do tempo são levadas em consideração, possibilitando a ocupação mais eficiente do canal e a justiça entre os fluxos. Além disso, todas se baseiam em algum dos modos de baixo consumo de energia para indicar aos possíveis mestres quando estarão ausentes de suas picorredes.

Dentre as propostas com decisão distribuída, Johansson et al. [51] apresentam

uma arquitetura para escalonamento em *scatternets*, centrada em um novo modo de funcionamento, o modo JUMP. Uma estação em modo JUMP em uma picorrede está, por padrão, ausente desta rede. Quando quiser estar presente em um enlace, precisa comunicar o fato ao seu par (mestre ou escravo). Isto fornece flexibilidade para a ponte alternar entre picorredes sem precisar comunicar o fato a cada salto. A arquitetura facilita o escalonamento interpicorrede, porém, a incorporação de um novo modo de funcionamento ao estado Conectado do controlador de enlace Bluetooth pode não ser viável. Os autores ainda não apresentaram simulações dessa proposta.

O trabalho de Kapoor et al. [13] apresenta outro mecanismo de decisão distribuída baseado no conceito de ponto de encontro (PE) acima definido. O mestre mantém uma lista dos PE das pontes com sua picorrede e uma lista dos PE de suas pontes com as outras picorredes a que pertencem. Os autores não deixam claro como essas informações são passadas ao mestre. De qualquer forma, essas informações permitem ao mestre otimizar a alocação de novos pontos de encontro. Deve-se lembrar que pontos muito próximos significam que, ao chegar em uma picorrede, a ponte já precisa sair, o que não é desejável. Esse algoritmo de alocação é simples, mas exige que muitas mensagens sejam trocadas para manter os mestres atualizados.

A proposta de escalonamento em topologia estruturada em árvore *Tree Scatternet Scheduling* (TSS) [41] foi elaborada para funcionar de forma integrada com o mecanismo de formação *Tree Scatternet Formation*, dos mesmos autores. Apesar dos autores alegarem que sua proposta de escalonamento funciona com outros algoritmos de formação de topologia, o emprego da topologia em árvore facilita o escalonador e torna mais fácil a obtenção de uma coordenação global entre as picorredes. Por outro lado, a aplicabilidade do algoritmo fica restrita a algumas topologias específicas, sem ciclos.

Finalmente, como última proposta com decisão distribuída, Baatz et al. [12] utilizam o modo de baixo consumo *Sniff* no escalonamento. A proposta se baseia em um esquema de crédito, inspirada no *deficit round robin*, em que cada estação estabelece um valor de crédito para os seus enlaces. Quando não houver dados a transmitir em um enlace, a estação distribui o crédito desse enlace entre os restantes. Uma estação transmite somente quando possui créditos. O modo *sniff* foi elaborado para funcionar de forma periódica e não com tráfegos que variam dinamicamente. Para permitir

a adaptabilidade do algoritmo, os parâmetros do *Sniff* precisam ser constantemente renegociados.

Dentre as propostas de mecanismos de escalonamento com decisão isolada, Racz et al. apresentam o algoritmo *Pseudo-Random Coordinated Scatternet Scheduling* (PCSS) [14]. Cada estação escolhe, pseudo-aleatoriamente, um ponto de encontro, baseado no relógio do mestre e no endereço do escravo. Como o par mestre-escravo usa os mesmos parâmetros no cálculo, ambos obterão os mesmos pontos de encontro. Estes, por sua vez, são diferentes para cada par. Quando as estações de um enlace se encontram, elas se comunicam até que uma delas decida atender a outro ponto de encontro. A principal vantagem do PCSS é a coordenação alcançada entre as estações sem a necessidade de sinalização explícita. Entretanto, com o aumento da quantidade de estações, os pontos de encontro, calculados aleatoriamente, começam a coincidir. Uma estação pode esperar desnecessariamente por seu par, que estará se comunicando em outro enlace.

Har-Shai et al. propõem o *Load Adaptive Algorithm* (LAA) [16] para ser executado apenas nas pontes. O algoritmo utiliza o modo *Hold* para que a ponte se ausente de uma picorrede e não requer modificações na especificação Bluetooth. A ponte se adapta a variações no tráfego observando suas filas de saída e recebendo a informação das filas dos mestres em sua direção. Para que o mestre passe a informação de sua fila em direção à ponte, é necessário um campo adicional nos pacotes. O trabalho não deixa claro como essa informação é passada. De qualquer forma, quando o tamanho de uma das filas observadas pela ponte ultrapassar um determinado limite, a ponte tentará trocar de picorrede. No estágio atual, o LAA é aplicável somente a pequenas *scatternets*, nas quais as pontes só podem estar conectadas a dois mestres.

Zhang e Cao [15] apresentam um algoritmo similar ao anterior. As pontes armazenam uma tabela contendo a seqüência de picorredes por onde vão passar. Uma réplica dessa tabela é mantida nos mestres, apenas para que eles saibam quando as pontes participam de suas picorredes. A proposta inicial prevê tabelas estáticas, definidas logo que a *scatternet* é formada. O algoritmo foi modificado para que as tabelas possam ser atualizadas conforme a dinâmica do tráfego, aumentando a complexidade do algoritmo. O mestre não atualiza sua tabela por conta própria, precisando ser informado

pela ponte sobre cada mudança. Existe, ainda, a limitação de que cada picorrede pode ter no máximo duas pontes.

De um modo geral, a maioria das propostas apresenta as seguintes limitações. Normalmente, a ponte pode pertencer somente a duas picorredes e funciona sempre como escravo. A quantidade de estações de uma *scatternet* também é limitada. As propostas avaliam as métricas de vazão agregada das *scatternets* e de retardo dos pacotes. Apenas a proposta PCSS [14] preocupa-se em medir a eficiência da ponte em relação ao consumo de energia. Ainda assim, essa eficiência no consumo de energia é uma conseqüência do funcionamento do algoritmo e não uma métrica de desempenho que pode ser configurada como objetivo principal.

3.5 Resumo

Este capítulo levantou as principais questões envolvendo as *scatternets*: formação da topologia, encaminhamento de pacotes e escalonamento interpicorrede. O último tópico mereceu destaque por estar diretamente relacionado à proposta desta dissertação. Para cada questão abordada, foram levantados os principais trabalhos publicados.

As propostas de escalonamento interpicorrede foram divididas em mecanismos com decisão distribuída e com decisão isolada. Os mecanismos com decisão distribuída possibilitam a coordenação global entre as picorredes, às custas de sinalização adicional e de modificações na especificação Bluetooth. Por outro lado, os mecanismos com decisão isolada possuem como grande atrativo sua simplicidade, requerendo modificações apenas nas pontes, sem a necessidade de nova sinalização de controle. Por isso, os mecanismos com decisão isolada podem ser implementados em curto prazo e apresentam maior potencial para serem incorporados à especificação.

O próximo capítulo apresenta, em detalhes, os algoritmos de escalonamento intrapicorrede e interpicorrede propostos nesta dissertação.

Capítulo 4

Propostas de Escalonamento Intrapicorrede e Interpicorrede

4.1 Introdução

Este capítulo apresenta os algoritmos de escalonamento intrapicorrede e interpicorrede propostos na dissertação. A política de escalonamento intrapicorrede tem influência direta no desempenho dos tráfegos de uma picorrede. A injustiça no compartilhamento do enlace de uma picorrede e o grande retardo causado nos pacotes de um fluxo específico são exemplos de efeitos indesejáveis provocados por um mecanismo de escalonamento ineficiente. Já o escalonamento interpicorrede, abordado na Seção 3.4, está diretamente relacionado ao desempenho dos tráfegos entre picorredes.

A Seção 4.2 aborda o algoritmo de escalonamento intrapicorrede proposto. A Seção 4.3 apresenta em detalhes a proposta para o escalonamento interpicorrede. Finalizando o capítulo, a Seção 4.4 expõe as conclusões.

4.2 Escalonamento Intrapicorrede: Algoritmo DRR-CoS

Dois tipos de enlaces podem ser estabelecidos em uma picorrede, conforme abordado na Seção 2.4.2. O enlace orientado a conexão (SCO) é criado entre o mestre e um escravo, com a finalidade de garantir vazão fixa e retardo limitado para o tráfego. Já o enlace sem conexão (ACL) é compartilhado entre o mestre e todos os escravos, ficando a cargo do mestre o controle do acesso ao meio. O mecanismo utilizado pelo mestre

para controlar este acesso ao meio é chamado de política de escalonamento intrapicorrede. Como o enlace ACL foi projetado para atender ao tráfego de melhor esforço, a política de escalonamento intrapicorrede deve buscar a otimização da ocupação do canal e a justiça no compartilhamento do enlace. Os principais trabalhos na área foram apresentados na Seção 2.4.8 ([9], [10], [11], [30]).

O enlace SCO garante uma vazão de 64kbps entre o mestre e um escravo, através da reserva de *slots* em intervalos fixos. O enlace ACL se restringe aos *slots* não reservados para o enlace SCO. Por exemplo, no caso de um tráfego de voz estabelecido com pacotes HV3 (ver Tabela 2.2), a cada 6 slots, dois são ocupados por estes pacotes, com a transmissão de 30 bytes de informação nos sentidos mestre-escravo e escravo-mestre. O tráfego de dados fica restrito aos 4 *slots* restantes, o que limita a comunicação a um pacote de 3 *slots*, com a resposta em um *slot*. Em termos de vazão, o tráfego de dados pode alcançar apenas 390,4kbps, contra os 723,2kbps que seriam possíveis com pacotes de 5 *slots*.

Dois outros fatores sugerem a busca de alternativas ao uso de enlaces SCO. O primeiro é a existência de codificadores ADPCM para voz que geram amostras a taxas de 16 a 40kbps e de codificadores recentes baseados em quadros que atingem taxas de 8kbps [52]. Conseqüentemente, os 64kbps garantidos pelo SCO tornam-se superdimensionados. O segundo fator é a existência de períodos de silêncio no tráfego de voz, tornando ainda maior o desperdício de banda provocado pelos enlaces SCO.

Este trabalho propõe um esquema de escalonamento que permite o compartilhamento de tráfego de voz e dados sobre um enlace ACL, garantindo, ainda assim, um retardo limitado para a voz. O algoritmo proposto para realizar o escalonamento intrapicorrede baseia-se no *deficit round robin* (DRR) [48] e denomina-se de DRR-CoS (*deficit round robin with classes of service*).

O DRR tenta garantir justiça no escalonamento atribuindo um *quantum* para as estações. O *quantum* funciona como um crédito, que é distribuído para as estações a cada rodada. Uma estação, ao ser escalonada, somente poderá transmitir se tiver crédito suficiente e continuará transmitindo até o término do seu crédito. A parte do crédito não utilizada em uma rodada é acumulada para a rodada seguinte.

O algoritmo DRR-CoS executado pelo mestre de uma picorrede define um intervalo entre escalonamentos sucessivos dos tráfegos com restrição de retardo através do parâmetro **IPol** (intervalo de *polling*). Com o intuito de configurar o valor deste parâmetro, o mestre deve manter uma relação das conexões L2CAP que apresentam restrição de retardo. Apenas um valor de **IPol** é armazenado. Portanto, mesmo que os fluxos apresentem limites de retardo diferentes, deve-se escolher o menor dos valores para **IPol**, a fim de se garantir a QoS para todos os fluxos. Possivelmente, o uso de um parâmetro para cada fluxo evitaria, algumas vezes, o escalonamento desnecessário de certas estações. Entretanto, a adoção de um único valor para todos os fluxos torna mais simples a implementação do algoritmo, bastando o mestre manter uma fila de estações com fluxos com restrição de retardo e percorrer esta fila a cada **IPol** segundos.

O DRR-CoS funciona da seguinte forma. O mestre mantém um contador, iniciado com o valor **IPol** e decrementado com o tempo. Quando seu valor chega a zero, o mestre passa a escalonar, sucessivamente, os escravos pertencentes à relação das conexões com restrição de retardo. Quando todos estes escravos tiverem sido escalonados, o mestre volta à política de escalonamento DRR tradicional, continuando do ponto onde havia interrompido.

Ao invés de garantir intervalos entre escalonamentos exatos como o SCO, o DRR-CoS fornece um intervalo próximo do valor de **IPol**. O valor é aproximado pois, esgotado o tempo **IPol**, o mestre espera o término da transmissão corrente, antes de iniciar o escalonamento das estações pertencentes à lista de fluxos com retardo limitado. No pior caso, a transmissão atual pode ocupar 10 *slots* (5 *slots* em cada sentido), permitindo uma variação no retardo do tráfego de até 6,25ms. Portanto, para se garantir que o limite de retardo estabelecido para um tráfego nunca seja extrapolado, deve-se configurar o **IPol** levando-se em consideração essa pequena variação. Deve-se ressaltar que dependendo do tamanho da fila de estações com fluxos com restrição de retardo, a variação do retardo dos pacotes pode ser aumentada.

A política de escalonamento intrapicorrede é independente do escalonamento interpicorrede. Entretanto, caso o mestre da picorrede tenha conhecimento sobre as estações pontes em sua rede, ele pode otimizar o escalonamento intrapicorrede, adotando duas medidas. Na primeira, se a ponte informar ao mestre o período de sua ausência da rede,

este pode retirá-la temporariamente da sua lista de escalonamento. Assim, o mestre pode dedicar-se integralmente aos escravos que permanecem na picorrede, evitando o desperdício de banda. Na segunda, como a ponte permanece uma parte do tempo ausente da picorrede, o mestre pode compensar esta ausência, escalonando-a, quando presente, mais freqüentemente do que os demais escravos.

O algoritmo DRR-CoS pode, facilmente, incorporar as funcionalidades supracitadas. A primeira pode ser implementada através da sinalização referente a um dos modos de baixo consumo de energia, no qual a ponte informa ao mestre o período de seu afastamento. Este assunto é abordado na seção seguinte. Já a segunda funcionalidade, que trata da diferenciação entre o escalonamento da ponte e dos demais escravos, é obtida definindo-se dois valores de *quantum*. O maior valor é atribuído à ponte e o menor valor aos demais escravos, garantindo-se que a ponte será escalonada mais vezes durante o período de sua presença na picorrede.

4.3 Escalonamento Interpicorrede: Algoritmo AISA

Existem algumas características desejáveis para os algoritmos de escalonamento interpicorrede. A simplicidade de implementação, traduzida pelo mínimo de alterações necessárias na especificação Bluetooth, é uma das mais importantes. Uma forma de se alcançar essa simplicidade de implementação é limitando as decisões sobre o escalonamento a algumas estações da rede, mais especificamente às pontes. Assim, o processamento fica restrito a essas estações, evitando-se a troca de mensagens de escalonamento entre estações vizinhas. A classificação proposta na Seção 3.4.1 coloca os algoritmos com essa característica na categoria de mecanismos de escalonamento com *decisão isolada*.

Em muitas situações, o algoritmo de escalonamento interpicorrede deve buscar, também, a otimização do desempenho do tráfego interpicorrede, que pode ser definida como a maximização de uma métrica de desempenho considerada prioritária, com o mínimo de prejuízo para as demais. A categoria de mecanismos de escalonamento com *decisão distribuída* pode alcançar uma eficiência maior que os com *decisão isolada*, devido à possibilidade de coordenação entre as pontes e os mestres das picorredes.

Entretanto, independentemente do mecanismo utilizado, o desempenho do tráfego interpicorrede é naturalmente pior do que o do tráfego interno a uma picorrede, uma vez que a ponte divide seu tempo entre as picorredes.

O algoritmo de escalonamento interpicorrede proposto nesta dissertação, chamado de AISA (*Adaptive Interpiconet Scheduling Algorithm*), é executado somente nas pontes, às quais cabem as decisões sobre a forma de alternância entre as picorredes. Portanto, o AISA encontra-se na categoria dos mecanismos de escalonamento de *decisão isolada*. O fato de minimizar as modificações na especificação Bluetooth foi um fator fundamental nesta escolha.

O AISA é um mecanismo de escalonamento adaptativo e parametrizável, que se diferencia das demais propostas de escalonamento interpicorrede, por permitir a escolha da métrica de desempenho que se deseja melhorar. As métricas compreendem a vazão agregada do tráfego, o retardo médio dos pacotes e o consumo de energia da ponte. Mesmo no caso das duas primeiras métricas, o algoritmo possui algumas características de implementação que possibilitam um menor consumo de energia comparado aos algoritmos de escalonamento tradicionais como o *round robin* e o *deficit round robin*. Além disso, o escalonador tenta garantir a justiça entre tráfegos de características similares.

A ponte escalona as suas picorredes de uma forma similar ao *weighted round robin* [49]. A ponte divide o tempo em turnos de duração fixa. Em cada turno, a ponte passa exatamente uma vez em todas as picorredes, permanecendo um tempo pré-determinado em cada uma delas. O cálculo do tempo de permanência em cada picorrede é obtido através da medição da ocupação do canal. Por sua vez, a medição da ocupação do canal é feita através da contabilização dos pacotes POLL e NULL nos sentidos mestre-escravo e escravo-mestre, respectivamente, e é atualizada a cada turno. Conforme a ocupação do canal aumenta ou diminui, a ponte ajusta os tempos de permanência em cada picorrede. As picorredes com maior tráfego de ou para a ponte recebem uma parcela maior do turno, reduzindo-se, conseqüentemente, o tempo de permanência da ponte nas picorredes com menor tráfego.

As pontes se comunicam com o mestre de cada picorrede através da sinalização LMP prevista para o modo de baixo consumo HOLD. Quando a ponte está prestes a deixar a picorrede atual, a mesma envia um pacote LMP_Hold_Req para o mestre da rede (caso a

ponte seja o mestre, não precisa haver sinalização alguma). O LMP_Hold_Req informa o instante em que a ponte entrará em modo HOLD e a duração deste estado. Assim, o mestre fica ciente da ausência da ponte e retira-a do escalonamento intrapicorrede durante o período combinado. Para informar à ponte sobre a aceitação da mudança para o modo HOLD, o mestre responde com um pacote LMP_Accepted.

A próxima subseção justifica a emprego do modo HOLD no algoritmo AISA. Posteriormente, são introduzidos os parâmetros associados ao AISA. Esses parâmetros são mencionados na subseção seguinte, ao explicar o funcionamento do algoritmo de forma detalhada. Finalmente, a última subseção apresenta um exemplo numérico, ilustrando o funcionamento do AISA.

4.3.1 O Emprego do Modo HOLD

Após algumas tentativas de comunicação sem resposta com o escravo escalonado, o mestre o desconecta por abandono à picorrede. Entretanto, este escravo pode ser uma ponte que esteja se comunicando com outra picorrede no referido instante. Para que esta situação seja evitada, utiliza-se um dos modos de baixo consumo de energia (HOLD, SNIFF ou PARK) para desativar a comunicação do mestre com a ponte. Assim, o mestre, com quem a ponte entra em acordo sobre o modo de baixo consumo, não a escalonará durante o período combinado.

Dentre os modos de baixo consumo existentes, o modo HOLD apresenta o maior potencial para ser empregado pela ponte na troca de picorredes. Quando o canal de comunicação entre o mestre e a ponte é colocado em HOLD, a especificação Bluetooth não permite que o mestre escalone o escravo inativo até que o período de inatividade se encerre. Dessa forma, pacotes de controle não são transmitidos. Em comparação ao modo SNIFF, o modo HOLD apresenta um ciclo de trabalho menor, propiciando uma maior economia de energia. Como desvantagem do modo HOLD, pode-se citar a necessidade de sinalização LMP sempre que a ponte vai deixar uma picorrede, mesmo que os parâmetros sejam os mesmos da última negociação. Entretanto, em um mecanismo adaptativo como o AISA, o tempo de permanência da ponte em suas picorredes varia dinamicamente, requerendo a sinalização de qualquer maneira.

O modo PARK proporciona a maior economia de energia dos três modos. Ao entrar em PARK, o dispositivo libera seu endereço de rede (AM_ADDR), permitindo a entrada de uma outra estação na rede. Entretanto, o processo de liberação e re-aquisição do endereço é muito mais demorado que a negociação do HOLD. Além disso, para se manter sincronizada à picorrede, a ponte em PARK precisa escutar, periodicamente, pacotes de controle. Portanto, o modo PARK não é recomendado no caso em que a ponte alterna com bastante freqüência entre picorredes.

O modo SNIFF vem sendo empregado para a estação ponte em alguns trabalhos ([12], [13]). O consumo de energia do modo SNIFF é o maior dos três modos de baixo consumo. Considerando que os tráfegos em uma *scatternet* variam dinamicamente, os parâmetros do SNIFF precisam ser redefinidos freqüentemente entre a ponte e suas picorredes para atender aos requisitos desses tráfegos. Como não existe uma mensagem LMP para alterar o período dos *slots* de SNIFF, teria que ser feita, inicialmente, uma comunicação entre a ponte e o mestre para a ponte voltar ao modo ATIVO e, posteriormente, outra sinalização para a ponte entrar no modo SNIFF com o novo período.

Uma vez adotado o modo HOLD, duas considerações precisam ser feitas. Apesar da especificação Bluetooth permitir que a solicitação de modo HOLD seja iniciada tanto pelo mestre quanto pelos escravos, no caso do AISA, a negociação será iniciada sempre pelo escravo, ou seja, a ponte (como mestre, não haveria necessidade do HOLD). Esta primeira consideração deve-se ao fato de que a ponte gerencia todos os encontros com as suas picorredes.

A segunda consideração trata do momento em que a solicitação de entrada em modo HOLD deve ser enviada. A especificação Bluetooth define um intervalo mínimo entre a solicitação de modo HOLD e a efetiva entrada neste modo. Esse intervalo mínimo é necessário, porque a ponte pode gerar o pacote LMP_Hold_Req, de solicitação de modo HOLD, durante a transmissão de outro pacote, atrasando o envio daquele pacote. Além disso, o mestre precisa ter tempo para responder com um LMP_Accepted. No caso do AISA, o parâmetro **sw_thresh** indica o momento em que a ponte deve gerar o pacote de solicitação LMP_Hold_Req. O instante **sw_thresh** ocorre um certo tempo antes do término do encontro da ponte com a picorrede.

4.3.2 Parâmetros do AISA

As principais variáveis envolvidas no código do AISA são descritas nesta seção. As variáveis são divididas em três grupos: contadores, registros para cada picorrede e parâmetros configuráveis. Os contadores mantêm informações sobre a picorrede atual da ponte. Os registros armazenam os valores específicos para cada picorrede ao longo do tempo. Os parâmetros são configurados antes da criação da *scatternet* para as futuras pontes. Os valores assumidos pelos parâmetros influem diretamente no funcionamento do AISA, beneficiando ou prejudicando determinada métrica de desempenho.

Os contadores são os seguintes:

- **turn_cnt** – iniciado com o valor **turn_sz**, registra o tempo restante no turno;
- **curr_pico_cnt** – registra o tempo restante da ponte na picorrede atual;
- **curr_util** – armazena a quantidade de *slots* usados na comunicação entre a ponte e a picorrede atual;
- **unused_slots** – armazena o total de *slots* liberados pelas picorredes;
- **reserved_slots** – caso a ponte não escalone uma determinada picorrede em um turno, os *slots* destinados a esta picorrede ficam reservados para o próximo turno.

Para cada picorrede associada à ponte, é criado um registro com as seguintes informações:

- **pico_id** – contém o identificador único para a picorrede;
- **next_meet** – armazena o instante do próximo encontro da ponte com a picorrede;
- **next_dur** – armazena a duração do próximo encontro da ponte com a picorrede;
- **avg_util** – registra a utilização média da picorrede.

A utilização média de cada picorrede é calculada pela média ponderada móvel, através da fórmula: $\mathbf{avg\_util} = \alpha \times \mathbf{avg\_util} + (\mathbf{1} - \alpha) \times \mathbf{curr\_util}$, onde α é um valor entre 0 e 1, representando a influência do passado no cálculo da média atual. Conforme α se aproxima de 1, as variações em **avg_util** ocorrem mais lentamente.

A configuração dos seguintes parâmetros permite otimizar uma determinada métrica de desempenho:

- **turn_sz** – armazena a duração do turno (em *slots* de 625μs);
- **min_dur**, **max_dur** – registram, respectivamente, os limites mínimo e máximo de permanência da ponte em uma picorrede;
- **inc_bound, dec_bound** – respectivamente, acima e abaixo destes limites, permite-se que o tempo de permanência da ponte em uma picorrede seja aumentado e diminuído;
- **inc_rate, dec_rate** – armazenam, respectivamente, as taxas de acréscimo e decréscimo do tempo de permanência da ponte em uma picorrede;
- **sw_thresh** – limite após o qual a ponte precisa enviar um pacote LMP_Hold_Req ao mestre da rede, indicando que está para deixar a picorrede atual;
- **early_exit** – se não houver mais dados a transmitir (indicado por uma transmissão de pacotes POLL/NULL) e ainda restar uma quantidade de *slots* igual ou superior a **early_exit** na picorrede atual, a ponte pode antecipar sua saída da rede, propiciando economia de energia;
- **skip_pico** – variável booleana que autoriza a retirada de uma picorrede do escalonamento em um turno, por não ter dados a transmitir, regressando no turno seguinte.

4.3.3 Funcionamento do Algoritmo AISA

As rotinas executadas em cada ponte [1] de uma *scatternet* são explicadas nesta seção. Basicamente, existem as rotinas executadas uma única vez quando a ponte deixa uma picorrede e aquelas executadas ao longo do período de permanência da ponte na picorrede. As rotinas são apresentadas, resumidamente, na forma de pseudo-código na Figura 4.1. O código contido na figura serve de base para as explicações contidas nesta seção e, por isso, suas linhas foram numeradas para facilitar sua referência.

O AISA foi projetado com a intenção de possibilitar a economia de energia das pontes. Esta economia ocorre quando a ponte entra, efetivamente, em modo de baixo consumo HOLD. Existem três possibilidades para a ponte entrar em HOLD.

[1] Todas as explicações desta seção consideram a ponte como um escravo nas picorredes em que participa. Caso a ponte funcione como mestre, ela tem o controle da picorrede e não precisa sinalizar sua saída.

```
Saindo de uma picorrede:

1-  Executar atualizar_pico_info;
2-  Se ([(todas pico escalonadas) E (turn_cnt != 0)] Ou (curr_pico_cnt >= early_exit))
    Então
3-      Executar modo HOLD;
4-  Senão
5-      Carregar info próxima pico.

Durante o período de permanência em uma picorrede:

6-  Se (pico for escalonada) Então
7-  Executar receber_pacote;
8-  Se (curr_pico_cnt <= sw_thresh) Então
9-      Criar LMP_Hold_Req (curr_pico_cnt, next_meet);
10- Executar enviar_pacote.

Rotina receber_pacote:

11-  Escolher (tipo de pacote recebido):
12-      Dados: Enviar pacote para a camada superior (L2CAP);
13-             Parar;
14-      Poll: Se (fila de saída para a estação está vazia) Então
15-              Se (curr_pico_cnt > early_exit) Então
16-                  Criar LMP_Hold_Req (saída antecipada, next_meet);
17-      Default: Receber pacote LMP.

Rotina atualizar_pico_info:

18-  Calcular avg_util;
19-  Se (avg_util > last_avg_util) Então
20-      Se (avg_util > inc_bound) Então
21-          Calcular acréscimo de slots X através de inc_rate,
             com a condição next_dur <= max_dur;
22-      Se (X < unused_slots) Então
23-          next_dur = next_dur + X;
24-      Senão
25-          Utilizar unused_slots;
26-          Obter pico com maior next_dur;
27-          Tentar transferir (X – unused_slots) slots para pico atual;
28-  Senão
29-      Se (avg_util < dec_bound) Então
30-          Calcular decréscimo de slots Y através de dec_rate,
             com a condição next_dur >= min_dur;
31-          Se (Y == 0) E (skip_pico está habilitado) Então
32-              Pico não será escalonada no próximo turno;
33-          Atualizar unused_slots;
```

Figura 4.1: Pseudo-código do funcionamento do AISA.

Na primeira, conforme o tráfego das picorredes com a ponte diminui, o tempo de permanência da ponte nestas também diminui e os *slots* liberados são acumulados em **unused_slots**. Como o turno tem duração fixa (**turn_sz**), podem sobrar *slots* no turno após o escalonamento de todas as picorredes. Durante este período, a ponte entra em modo de baixo consumo HOLD até o término do turno, quando **turn_cnt** é igual a zero. Isto está representado pela duas primeiras condições da linha 2 da Figura 4.1.

A segunda possibilidade de economizar energia é caracterizada pela saída antecipada da ponte de uma picorrede, o que ocorre quando a ponte e seu par na comunicação não têm mais dados a transmitir. Como existe uma perda de *slots* com a negociação do HOLD, a saída antecipada da ponte de uma picorrede só é compensatória se o tempo restante na picorrede (**curr_pico_cnt**) for maior que um determinado limite (**early_exit**). Esta parte do algoritmo está representada nas linhas 14, 15 e 16 e na terceira condição da linha 2 do pseudo-código.

A terceira possibilidade ocorre quando a ponte já permanece durante o tempo mínimo (**min_dur**) permitido em uma picorrede e, ainda assim, a ocupação do canal está baixa. Nesse caso, a ponte pode deixar de escalonar essa picorrede no próximo turno, voltando a escaloná-la no turno seguinte ao próximo. A ponte nunca deixa de escalonar a mesma picorrede em dois turnos consecutivos. Para permitir esta forma de economia de energia, é necessário que o parâmetro **skip_pico** esteja habilitado. Esta parte do código está caracterizada entre as linhas 30 e 32 do pseudo-código.

Ao sair de uma picorrede, a ponte precisa atualizar as informações sobre o próximo encontro com a mesma. Esta atualização é feita através da rotina *atualizar_pico_info*, explicada mais adiante nesta seção. Em seguida, a ponte pode entrar em modo HOLD, conforme explicado anteriormente, ou comunicar-se com outra picorrede. No segundo caso, a ponte carrega o registro com as informações sobre a referida picorrede, incluindo a duração da comunicação (**next_dur**) e a utilização média do enlace (**avg_util**). O código deste parágrafo corresponde às linhas 1 a 5 da Figura 4.1.

Durante o funcionamento normal de uma picorrede, sempre que a ponte é escalonada, ela verifica se já está na hora de deixar a rede. Esta condição é verificada na linha 8 do pseudo-código. Se for o caso, a ponte envia um pacote LMP_Hold_Req para o mestre, informando o instante em que deixará a picorrede (daqui a **curr_pico_cnt**)

e quando será o próximo encontro entre ambos (**next_meet**). Portanto, o **next_meet** precisa ser calculado antes da chamada à rotina *atualizar_pico_info*.

O cálculo do próximo encontro (**next_meet**) é feito da seguinte maneira. A primeira picorrede escalonada pela ponte em um turno terá o seu próximo encontro no início do turno seguinte. O próximo encontro da segunda picorrede com a ponte ocorrerá logo após o término da primeira, ou seja, $\mathbf{next\_meet_{segunda}} = \mathbf{next\_meet_{primeira}} + \mathbf{next\_dur_{primeira}}$. Mesmo que uma picorrede deixe de ser escalonada em um turno, o cálculo de **next_meet** para a picorrede seguinte leva em consideração o tempo reservado para sua antecessora.

A rotina *atualizar_pico_info* é a parte de maior processamento do AISA e é executada sempre que a ponte deixa uma picorrede. Inicialmente, atualiza-se a utilização média **avg_util**, a partir da fórmula definida na Seção 4.3.2. O valor anterior a esta atualização, chamado de **last_avg_util** na linha 19 do pseudo-código, é guardado temporariamente para fins de comparação. Dependendo do resultado desta comparação, o algoritmo pode aumentar ou diminuir a duração do próximo encontro da ponte com a picorrede.

Para que a duração do próximo encontro (**next_dur**) entre a ponte e a picorrede seja aumentada, é necessário um acréscimo na utilização média (**avg_util**) e esta deve estar acima do limite **inc_bound**. Se ambas as condições forem satisfeitas, a taxa de acréscimo de *slots* (**inc_rate**) é empregada para calcular a nova duração do encontro. De posse da taxa **inc_rate**, ainda precisa ser verificado se existem *slots* desocupados suficientes no turno e se a duração resultante não é maior que a duração máxima permitida (**max_dur**). No caso da primeira condição, se houver *slots* livres suficientes (parâmetro **unused_slots**), a ponte retira a quantidade necessária, incorporando-a à duração do seu próximo encontro (**next_dur**) com a picorrede correspondente. Caso contrário, a ponte tenta retirar *slots* da picorrede com o maior **next_dur**. A picorrede da qual se retiram *slots* não pode ficar com a duração menor do que a da solicitante. Essas condições ajudam a manter a justiça na distribuição do turno e evita-se que uma picorrede com **next_dur** longo detenha esta parcela de tempo indefinidamente. Esta parte do código está contida entre as linhas 21 a 27.

A duração do próximo encontro entre a ponte e a picorrede atual pode ser dimi-

nuída, se houver redução na utilização média do canal existente entre ambas e se esta utilização estiver abaixo do limite **dec_bound**. Em caso afirmativo, a ponte utiliza a taxa de decréscimo de *slots* (**dec_rate**) para calcular de quanto será reduzido o próximo encontro. Caso o encontro atual já esteja ocupando a duração mínima (**min_dur**), a ponte deixará de escalonar esta picorrede no próximo turno, conforme explicado anteriormente, na terceira possibilidade para a ponte entrar em modo HOLD. Este trecho do código está descrito entre as linhas 29 e 33.

Em resumo, os parâmetros citados nesta subseção foram criados com as seguintes finalidades. O limite **inc_bound** visa garantir que a ponte só aumente a duração de um encontro, se a ocupação do canal estiver alta, tornando a duração desses encontros mais estável. Analogamente, o limite **dec_bound** também visa tornar a duração dos encontros mais estável, pois a ponte só diminui sua duração, se a ocupação do canal estiver baixa. As taxas de acréscimo (**inc_rate**) e decréscimo (**dec_rate**) de *slots*, em conjunto com os dois parâmetros anteriores, visam aumentar a adaptabilidade da ponte às condições do tráfego. A duração máxima (**max_dur**) pode ser configurada para evitar que uma picorrede ocupe a maior parte do turno. Isto pode ser interessante, por exemplo, para deixar um período do turno livre para que a ponte economize energia. A duração mínima (**min_dur**) tende a evitar a troca muito freqüente da ponte entre as picorredes, o que causaria o desperdício de *slots*. Os parâmetros **skip_pico** e **early_exit** visam possibilitar a economia de energia da ponte.

O efeito dos parâmetros do AISA será analisado através de simulações no Capítulo 5.

4.3.4 Exemplo do Funcionamento do AISA

Esta seção apresenta um exemplo numérico do funcionamento do algoritmo de escalonamento proposto. Foram adotados os seguintes parâmetros:

- **turn_sz** = 120 slots (40 slots/picorrede no início da simulação);
- **inc_bound** = 80
- **dec_bound** = 60
- **inc_rate** = 5 slots (acréscimo aditivo);

- **dec_rate** = 10 slots (decréscimo subtrativo);
- **max_dur** = 80 slots;
- **min_dur** = 20 slots;
- **early_exit** = 15 slots;
- **skip_pico** = 0.

Nesta configuração, a ponte pode sair antecipadamente de uma picorrede caso haja 15 ou mais *slots* restantes (**early_exit**) na picorrede e não haja mais dados a transmitir. Nenhuma picorrede pode deixar de ser escalonada pela ponte, pois **skip_pico** é zero. Todas as picorredes iniciam com a ocupação total de seus enlaces, ou seja, **avg_util** = **1**.

O valor **avg_util** está sendo calculado com a fórmula **avg_util** = $\mathbf{0,5} \times$ **avg_util** + $\mathbf{0,5} \times$ **curr_util**, ou seja, **avg_util** é a média aritmética entre a utilização neste turno e a média anterior.

Supõe-se que uma *scatternet* composta por três picorredes, interligadas por uma estação ponte, já esteja formada. Cada picorrede inicia com 40 slots de presença da ponte. A Tabela 4.1 mostra a evolução da ponte ao longo do tempo. Todos os valores de tempo na tabela são expressos em termos de *slots*.

Valores usados no turno atual				**Resultados ao fim de cada picorrede**				
Tempo	**Pico**	**next_dur**	**Saída antecipada**	**curr_util**	**avg_util**	**next_dur**	**next_meet**	**unused_slots**
0	1	40	0	0,80	0,90	40	120	0
40	2	40	20	0,50	0,75	40	160	0
80	3	40	20	0,50	0,75	40	200	0
120	1	40	0	0,80	0,85	40	240	0
160	2	40	30	0,25	0,50	30	280	10
200	3	40	25	0,35	0,55	30	310	20
240	1	40	0	0,95	0,90	45	360	15
280	2	30	0	0,60	0,55	30	405	15
310	3	30	0	0,65	0,60	30	435	15
340	hold	20						
360	1	45	0	0,94	0,92	50	480	10
405	2	30	15	0,49	0,52	20	530	20
435	3	30	0	0,70	0,65	30	560	20
465	hold	15						

Tabela 4.1: Exemplo numérico do funcionamento do AISA.

A leitura da tabela deve ser feita da seguinte maneira. Por exemplo, no tempo zero, a ponte está conectada à picorrede 1, onde deve permanecer por 40 *slots* (**next_dur** previsto para o turno atual). A ponte não saiu antecipadamente da picorrede (saída antecipada = 0), o que só seria permitido se houvesse uma transmissão de pacote POLL no sentido mestre-escravo, com a resposta NULL e ainda restasse **early_exit** slots na picorrede. A utilização neste turno foi 0,80 (**curr_util**). Como todas as picorredes iniciaram com **avg_util** = 1, a utilização média calculada foi de 0,90. Apesar de **avg_util** estar acima do limite **inc_bound**, não houve aumento da utilização (houve redução de 1 para 0,9). Por isso, a duração do próximo encontro (**next_dur**) continua igual a 40. Este encontro ocorrerá no tempo 120 (**next_meet**) e não há *slots* livres (**unused_slots** = 0).

No tempo 160, a ponte saiu antecipadamente da picorrede quando ainda restavam 30 *slots* para o fim do encontro. Como a utilização média (**avg_util**) ficou abaixo do limite **dec_bound**, a duração do próximo encontro (**next_dur**) será reduzida de 10 *slots* (valor de **dec_rate**), totalizando 30 *slots*. Em contrapartida, **unused_slots** foi incrementado de 10 *slots*. A variável **unused_slots** acumulará mais 10 *slots* ao final do encontro com a picorrede 3, iniciado no tempo 200.

O número de *slots* não utilizados ao final de um turno indica o tempo em que a ponte permanecerá em modo HOLD no turno seguinte. Por exemplo, no início do turno em 240, **unused_slots** é 20 e, portanto, após o escalonamento das três picorredes, sobrarão 20 *slots* no turno. No tempo 340, a ponte entra em HOLD, retornando à atividade em 360. Repare que no tempo 340, **unused_slots** é 15, mas o tempo de HOLD é o valor no início do turno.

Se no primeiro turno alguma picorrede tentasse aumentar sua próxima duração (**next_dur**), isto não seria possível, pois todas tinham a mesma duração e não havia *slots* livres. Já na saída da picorrede 1, próximo ao tempo 280, a ponte pôde aumentar a duração para 45, pois havia *slots* não utilizados. Finalmente, se no final da picorrede 2, próximo ao tempo 435, a ponte tentasse aumentar **next_dur** e não houvesse *slots* livres, a ponte transferiria 5 *slots* (equivalente a **inc_rate**) da picorrede 1 para 2. Isto visa manter a justiça na distribuição dos *slots*.

4.4 Resumo

Este capítulo descreveu os algoritmos de escalonamento intrapicorrede, chamado DRR-CoS, e interpicorrede, chamado AISA, propostos na dissertação. O DRR-CoS possibita o compartilhamento do enlace ACL entre o tráfego de melhor esforço e o tráfego com requisito de retardo limitado, sem a degradação deste último. Até então, as principais propostas de escalonamento intrapicorrede limitavam-se ao tratamento dos fluxos de melhor esforço, mantendo fluxos tais como voz em enlaces SCO. O capítulo seguinte mostra que o DRR-CoS é viável e melhora o desempenho da picorrede.

O AISA diferencia-se das demais propostas de escalonamento interpicorrede, por permitir a escolha da métrica de desempenho que se deseja enfatizar. Esta métrica pode ser otimizada a partir da configuração de determinados parâmetros do algoritmo. Um dos enfoques principais do algoritmo é a economia de energia e, mesmo em casos onde se busca aumentar a vazão agregada ou diminuir o retardo dos pacotes, é possível evitar o desperdício de *slots*, economizando energia. Os resultados do funcionamento do AISA são explorados no próximo capítulo.

Capítulo 5

Simulações e Resultados

5.1 Introdução

Este capítulo apresenta os principais resultados de simulações, realizadas com o objetivo de comprovar a eficácia dos algoritmos de escalonamento intrapicorrede e interpicorrede propostos.

A próxima seção descreve a ferramenta de simulação desenvolvida para o estudo de desempenho do Bluetooth. Esta ferramenta é uma extensão para o simulador de redes NS-2 [18] e foi empregada em todas as simulações descritas neste capítulo. A Seção 5.3 mostra os resultados relacionados ao algoritmo de escalonamento intrapicorrede DRR-CoS. Na Seção 5.4, são apresentados os principais resultados obtidos com o algoritmo de escalonamento interpicorrede AISA. A Seção 5.5 finaliza o capítulo, destacando seus pontos de maior relevância.

5.2 Ferramenta de Simulação BlueNetS

A técnica adotada para comprovar a eficácia dos mecanismos de escalonamento propostos foi a realização de simulações. Entretanto, até o início desta dissertação, havia apenas um simulador de Bluetooth disponível com código aberto: o BlueHoc [53], baseado no simulador de redes *Network Simulator* (NS-2) [18]. O BlueHoc implementa diversas funcionalidades do Bluetooth, mas, até o início deste trabalho, permaneciam algumas limitações da ferramenta como, por exemplo, tráfego apenas no sentido mestre-escravo, limite de apenas um tráfego por escravo e suporte a poucas aplicações. Quanto às *scatternets*, a ferramenta apresentava mais limitações. A estação ponte comutava

entre picorredes, sem, contudo, poder encaminhar pacotes entre as redes. Além disso, a ponte só podia assumir o papel de escravo.

Diversas alterações de código teriam que ser feitas para adaptar o BlueHoc [1] à proposta desta dissertação. No lugar de modificá-lo, optou-se pelo desenvolvimento de um novo simulador, também baseado nos recursos já existentes do simulador NS-2.

A ferramenta BlueNetS [2]. (*Bluetooth Network Simulator*) tem como objetivo simular o funcionamento das camadas física e de enlace da pilha Bluetooth, a partir do instante em que as conexões estejam estabelecidas, ou seja, a partir do início da transferência de dados. As seguintes funcionalidades, descritas na Seção 2.4, estão implementadas: duplexação por divisão no tempo (TDD), configuração de enlaces ACL, encaminhamento de pacotes nos sentidos mestre-escravo e escravo-mestre, mecanismos de correção de erros (FEC) e retransmissão (ARQ), segmentação e remontagem (SAR), modelo de erros provocados pela distância entre as estações, funcionamento da ponte nas configurações escravo/escravo e mestre/escravo e diversos algoritmos de escalonamento intrapicorrede e interpicorrede.

Com as funcionalidades citadas, BlueNetS forma a base necessária para o estudo de diversos mecanismos relacionados às camadas superiores da pilha Bluetooth como, por exemplo, o roteamento de pacotes, o comportamento do protocolo TCP e a descoberta de serviços. Como o BlueNetS baseia-se no simulador NS-2, o usuário não precisa ter um conhecimento profundo sobre a implementação do BlueNetS para criar, por exemplo, um módulo de roteamento para redes *ad hoc*, que possa ser aplicado sobre Bluetooth. Basta o usuário conectá-lo ao BlueNetS da mesma forma que ele o faz com um módulo de roteamento já existente no NS-2.

[1] O desenvolvimento do BlueHoc parece ter sido descontinuado, pois a ferramenta não foi atualizada nos últimos meses.

[2] Detalhes sobre a ferramenta BlueNetS, além dos expostos nesta dissertação, podem ser obtidos no trabalho [17] e através do envio de mensagens para o endereço eletrônico *werner@de9.ime.eb.br*.

5.2.1 Características do NS-2

O NS-2 [18] é um simulador de redes orientado a objetos, baseado em eventos discretos. Seu código é escrito em duas linguagens: C++ e OTcl. O simulador é formado por vários componentes, escritos em C++, os quais podem ser combinados para formar topologias de rede específicas. Os componentes incluem, entre outros, nós, aplicações, agentes, protocolos de roteamento, filas, enlaces, pacotes e temporizadores. A linguagem OTcl é usada como interface para a criação de cenários de simulação. Para cada classe C++, existe uma classe correspondente em OTcl. Assim, o usuário usa comandos OTcl para combinar os componentes desejados e formar o seu *script* de simulação.

A versão do NS-2 utilizada no desenvolvimento da ferramenta BlueNetS foi a 2.1b8a.

5.2.2 Modificações Realizadas no NS-2

A ferramenta BlueNetS aproveita ao máximo os módulos do NS-2 já existentes, modificando apenas o necessário para permitir a simulação do funcionamento do Bluetooth. A implementação foi realizada abstraindo-se a divisão em camadas da pilha Bluetooth.

Normalmente, em simulações de redes móveis com o NS-2, a inclusão de novas funcionalidades é feita na estrutura do componente nó móvel. A ferramenta BlueNetS seguiu outra abordagem, onde foram usados os módulos básicos de redes fixas para a criação das picorredes. O código se concentra no enlace unidirecional entre nós do NS-2 (classe *Simplex-link* do NS-2).

Durante uma simulação, o enlace entre duas estações pode ser desativado ou ativado, simulando uma queda ou restabelecimento de conexão, respectivamente. A alternância entre ativação e desativação dos enlaces unidirecionais pode ser empregada na simulação do TDD entre o mestre e um escravo. Por exemplo, em uma picorrede, todos os enlaces entre as estações permanecem desativados, exceto o enlace mestre-escravo escalonado pelo mestre naquele instante. Após a transmissão do mestre para o escravo, BlueNetS desativa o enlace mestre-escravo e ativa o escravo-mestre, possibilitando a transmissão

no sentido contrário. Esta e outras características do Bluetooth foram modeladas, alterando-se apenas a estrutura do enlace unidirecional do NS-2. A Figura 5.1 mostra os módulos que compõem o referido enlace após as modificações propostas.

Basicamente, o enlace unidirecional entre dois nós no NS-2 é composto dos módulos: **queue_**, simulando a fila de pacotes de saída do nó origem do enlace, **link_**, modelando as características de banda e retardo do enlace, **drophead_**, processando o descarte de pacotes do topo da fila e **ttl_**, verificando o *time to live* de cada pacote. Os módulos **enqT_**, **deqT_**, **drpT_** e **rcvT_** monitoram e registram, respectivamente, a entrada de pacotes na fila, sua saída da fila, seu descarte e seu recebimento pelo nó destino. A seguir, são destacadas as alterações realizadas no enlace.

O módulo **btmac_** é o responsável pela configuração do tipo de pacote banda base a ser utilizado na transmissão. A escolha do pacote pode ser feita pelo usuário através de um parâmetro de entrada ou pode-se deixar que o simulador a realize, mediante uma política de *best-fit*, escolhendo o tamanho de pacote que melhor se ajuste à quantidade de dados disponíveis. Caso o tamanho do pacote recebido da camada superior pelo L2CAP seja maior que a capacidade do pacote de banda base utilizado, o módulo **btmac_** realiza a segmentação do pacote e encaminha os segmentos resultantes para o módulo **queue_**.

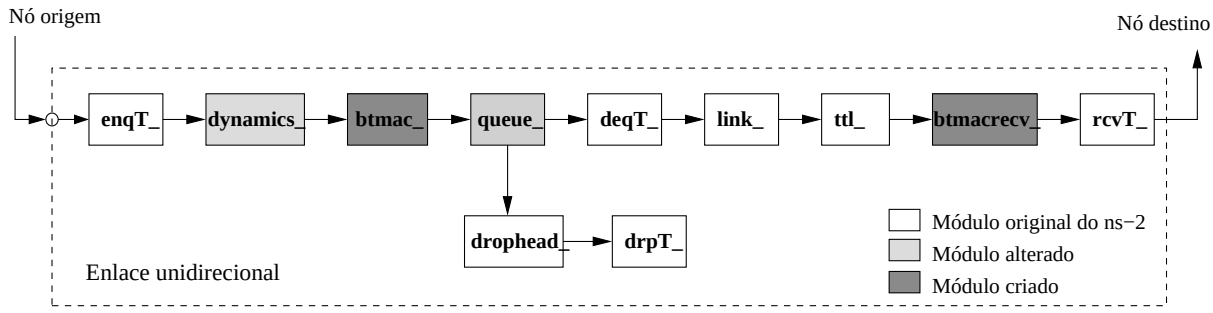


Figura 5.1: Componentes do enlace unidirecional NS-2 com modificações.

O módulo **queue_** simula a fila de pacotes de saída do nó (*host controller buffer*). No NS-2 sem modificações, quando um enlace é desativado, a fila é automaticamente esvaziada. No BlueNetS, a queda de um enlace indica que o nó correspondente não está sendo escalonado pelo mestre naquele instante. Portanto, modificou-se a estrutura de queue_ para que os pacotes não fossem descartados.

Nesse módulo, foram incluídos o modelo de erros e o mecanismo de ARQ. O modelo de erro implementado é o causado pela distância entre o transmissor e o receptor. Este modelo de erros foi adaptado do simulador BlueHoc [53], o qual é baseado no trabalho de Kumar e Gupta [54]. No momento da transmissão de um pacote, o modelo de erros é aplicado. Se a transmissão falhar, verifica-se a possibilidade de efetuar uma retransmissão. Se houver essa possibilidade, o pacote é mantido na fila, aguardando nova transmissão. Caso contrário, é descartado. Se o pacote descartado for um segmento de um pacote maior, todos os segmentos correspondentes também deverão ser descartados.

Os pacotes que atravessam o enlace com sucesso são recebidos pelo módulo **btmacrecv_**, o qual realiza a remontagem de pacotes. Este módulo acumula os segmentos de um pacote até o recebimento do último segmento, quando, então, encaminha o pacote completo para o próximo módulo.

Para fornecer maior flexibilidade ao simulador, a função de escalonamento foi deixada como uma extensão em OTcl. Assim, incorporam-se novas políticas de escalonamento sem a necessidade de recompilação do código do NS-2.

Uma desvantagem do BlueNetS está na execução lenta das simulações. A ferramenta é baseada na ativação e desativação dos enlaces, simulando o TDD. Como esses eventos são gerados a cada 625 microsegundos (tempo de um *slot*), a fila de eventos do NS-2 torna-se muito grande, causando a demora na execução.

Maiores detalhes sobre a ferramenta BlueNetS podem ser obtidos no trabalho [17], onde são apresentados resultados de validação da ferramenta, e também através do envio de mensagens para o endereço eletrônico *werner@de9.ime.eb.br*.

5.3 Escalonamento Intrapicorrede

Nesta seção, o algoritmo de escalonamento DRR-CoS será comparado com outras políticas de escalonamento intrapicorrede, com o objetivo de mostrar que o algoritmo proposto mantém o retardo limitado para o tráfego com requisito de retardo limitado, sem, contudo, degradar o tráfego de melhor esforço.

5.3.1 Cenário de Simulação

O cenário desta simulação é uma picorrede formada pelo mestre e sete escravos, conforme mostra a Figura 5.2. Todos os escravos estão distantes cerca de cinco metros do mestre, evitando-se a influência do erro de transmissão causado pela distância na simulação. Há dois fluxos de transferência de arquivos (FTP) para dois escravos distintos, com pacotes de tamanho fixo de 1000 bytes, usando transporte TCP New Reno. Há quatro fluxos bidirecionais de taxa constante (CBR), com pacotes de 512 bytes, transmitidos à taxa de 80 kbps. O fluxo com retardo limitado é o de voz, transmitido do mestre para um escravo. A voz foi estudada com duas modelagens diferentes.

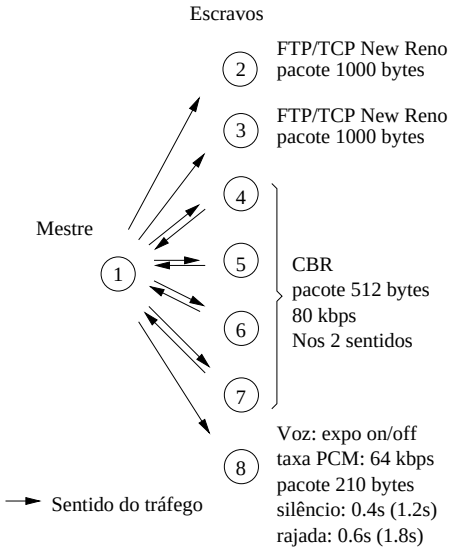


Figura 5.2: Cenário para teste do DRR-CoS.

Na primeira modelagem, a voz se comporta como um processo que alterna entre períodos de transmissão e silêncio, seguindo distribuições exponenciais com médias 0.4 seg e 0.6 seg, respectivamente [55]. Considerando que um codificador PCM gera dados a uma taxa de 64 kbps, pode-se dizer que no período de transmissão de voz, os pacotes, com tamanho de 210 bytes, chegam, em média, a cada 26ms. Por isso, o parâmetro **IPol** (intervalo de *polling*), que limita o retardo no algoritmo DRR-CoS, foi configurado para 20ms. Assim, os pacotes de voz tendem a ser servidos logo que chegam à fila de saída da estação, evitando-se o aumento do retardo pelo acúmulo de pacotes na fila.

Na segunda modelagem da voz, a taxa é mantida a 64 kbps, mas os períodos de tráfego e silêncio seguem distribuições exponenciais com médias 1,2 seg e 1,8 seg, res-

pectivamente [56], [52]. Apesar da proporção de tráfego/silêncio ser mantida inalterada em relação ao outro modelo, este apresenta períodos mais longos de tráfego e, portanto, rajadas mais longas.

Os algoritmos de escalonamento intrapicorrede comparados com o DRR-CoS foram: *Round Robin* (RR), *Exhaustive Round Robin* modificado (ERR-M), *Deficit Round Robin* (DRR) e *Deficit Round Robin* ideal (DRR-I).

Com o algoritmo RR, o mestre ordena os escravos e escalona-os uma vez, seqüencialmente. Após o atendimento ao último escravo, o mestre reinicia o escalonamento pelo primeiro escravo.

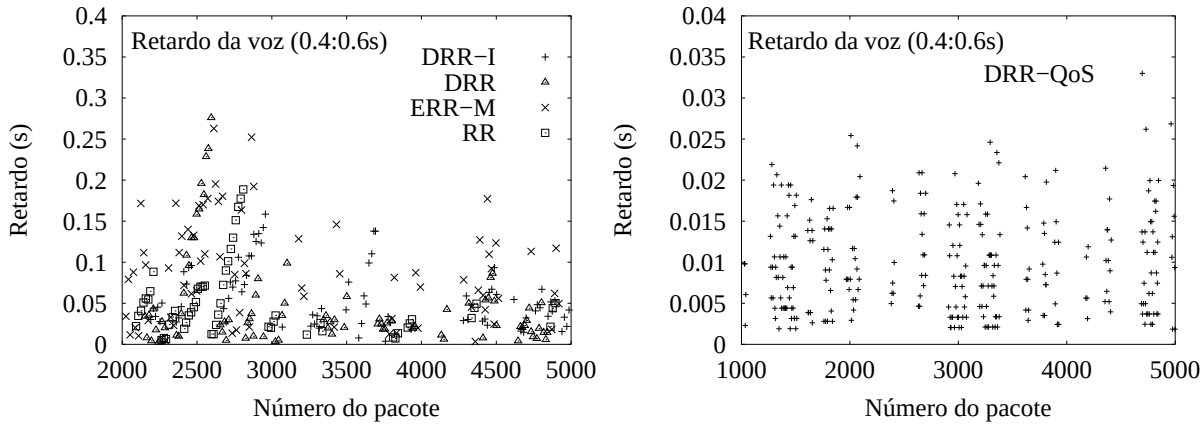
No algoritmo ERR-M, o mestre continua atendendo um mesmo escravo até que as filas em ambos os sentidos se esvaziem ou até que um limite (M) seja atingido, quando, então, o mestre escalona o próximo escravo, excetuando-se o atual, cuja soma das filas nos sentidos mestre-escravo e escravo-mestre seja a maior. A informação da fila do escravo só é obtida quando este é escalonado. O parâmetro M indica a quantidade máxima de pacotes que uma estação pode transmitir em uma rodada. Assim, evita-se que uma estação com muitos dados a transmitir retenha indefinidamente o direito de transmissão.

Quanto menor for o valor do parâmetro M, mais o funcionamento do ERR-M se aproxima do RR. Por outro lado, quanto maior for M, maior será o intervalo entre escalonamentos sucessivos do escravo com tráfego de voz, aumentando o retardo dos pacotes deste tráfego. Neste cenário, utilizou-se M igual a cinco pacotes. Se cada fluxo transmitir pacotes de 5 *slots*, esse valor de M representará a transmissão de até 25 *slots* consecutivos por escravo escalonado. Este valor tende a retardar bastante os escalonamentos do escravo com tráfego de voz, servindo ao propósito da simulação.

O DRR-I é uma implementação do DRR (explicado na Seção 4.2) em que o mestre conhece, a todo instante, suas filas e as dos escravos. Apesar de não ser uma implementação realista, serve como parâmetro de comparação. Tanto no DRR quanto em suas variantes, utilizou-se o *quantum* igual a oito *slots*. Este é um valor intermediário entre o RR, que escalona um pacote por escravo em cada rodada, e o ERR-M, que escalona até cinco pacotes por escravo.

5.3.2 Resultados do Escalonamento Intrapicorrede

O objetivo desta simulação é monitorar o retardo fim-a-fim dos pacotes de voz, obtido através das diversas políticas de escalonamento e compará-lo com o obtido através do DRR-CoS. O resultado comparativo do cenário com a primeira modelagem do tráfego de voz é mostrado na Figura 5.3, onde são apresentados os retardos de cada pacote de voz transmitidos em um período da simulação. A simulação teve duração de 120 segundos e foi executada dez vezes, mostrando, em todos os casos, resultados de retardo similares. Todos os resultados relacionados a médias foram calculados com intervalo de confiança de 95%. Para facilitar o entendimento desta figura, os resultados obtidos com o DRR-CoS estão separados no item (b) da Figura 5.3.



(a) Algoritmos ERR-M, DRR-I, DRR, RR

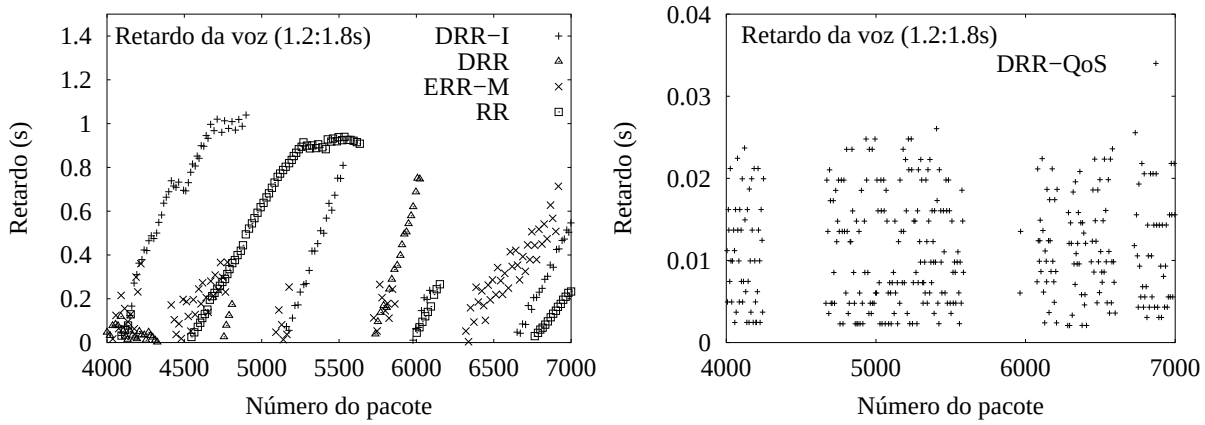
(b) Algoritmo DRR-CoS

Figura 5.3: Retardo dos pacotes de voz (tráfego: 0,4s; silêncio: 0,6s).

O resultado mostra que o retardo dos pacotes de voz chega a 200 ms em todos os mecanismos de escalonamento da Figura 5.3 (a), mas se mantém limitado a cerca de 25 ms ao longo de toda a simulação com o DRR-CoS.

No caso do cenário testado com a segunda modelagem do tráfego de voz, o resultado é mostrado na Figura 5.4. Neste caso, as médias das distribuições exponenciais dos períodos de tráfego e silêncio são 1,2 seg e 1,8 seg, respectivamente. Novamente, os resultados obtidos com o DRR-CoS estão separados na Figura 5.4 (b).

Como nesta modelagem da voz, o período médio de transmissão é mais longo do que na anterior, o retardo provocado nos pacotes de voz aumenta muito, chegando a



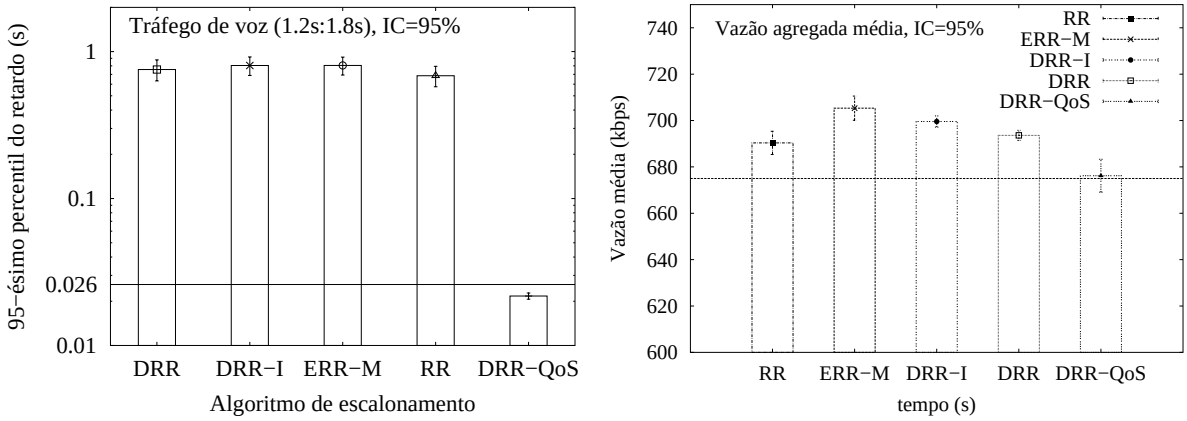
(a) Algoritmos ERR-M, DRR-I, DRR, RR (b) Algoritmo DRR-CoS

Figura 5.4: Retardo dos pacotes de voz (tráfego: 1,2s; silêncio: 1,8s).

um segundo. Novamente, o retardo dos pacotes usando o algoritmo DRR-CoS ficou limitado em cerca de 25ms, comprovando a eficácia do mecanismo. Ainda sobre o retardo dos pacotes, após o término de cada rodada de simulação, avaliou-se o 95-ésimo percentil do retardo. Ao final das dez rodadas, foi calculada a média dos resultados de percentil, com intervalo de confiança de 95%. O resultado é mostrado na Figura 5.5 (a). Enquanto o algoritmo DRR-CoS mantém o retardo dos pacotes abaixo de 26 ms, os outros algoritmos fornecem valores de retardo acima de 650 ms.

Vale ressaltar que, como os algoritmos usados na comparação com o DRR-CoS não foram criados com a preocupação de priorizar um determinado fluxo, era previsível que seus resultados em termos de retardo fossem (e realmente o foram) piores do que os obtidos com o DRR-CoS. Entretanto, esses algoritmos são muito importantes nas comparações em relação à métrica de vazão agregada. Um algoritmo com restrições temporais como, por exemplo, o *Earliest Deadline First* [57] poderia ter sido empregado nas simulações das duas métricas.

Também é importante verificar a influência do algoritmo DRR-CoS no comportamento do tráfego de dados da mesma picorrede. Para tanto, mediu-se a vazão agregada dos tráfegos de dados ao longo desta última simulação e calculou-se a média das dez rodadas, com intervalo de confiança de 95%. O resultado é apresentado na Figura 5.5 (b).



(a) Percentil do retardo dos pacotes

(b) Vazão agregada média

Figura 5.5: Resultados para o tráfego de voz (tráfego: 1,2s; silêncio: 1,8s).

A vazão agregada média obtida com o DRR-CoS, próxima de 680 kbps, é um pouco inferior à obtida com os outros algoritmos de escalonamento. Esta diferença ocorre porque o mecanismo DRR-CoS escalona a estação com o tráfego de voz a cada 20 ms. Como nem sempre há pacotes a transmitir por esta estação, alguns *slots* são desperdiçados.

Comparativamente ao emprego do tráfego de melhor esforço em conjunto com um enlace SCO, projetado a atender ao tráfego com requisito de retardo limitado, a vazão agregada propiciada pelo algoritmo DRR-CoS foi muito superior. Se existir um enlace SCO com pacotes HV3, a vazão teórica máxima para os dados de melhor esforço é de 390,4 kbps, mais os 64 kbps reservados para a voz. O uso de pacotes HV2 deixa apenas dois *slots* livres entre os reservados para a voz, e com HV1, não há espaço para a transmissão de dados.

5.4 Escalonamento Interpicorrede

Nesta seção, os resultados obtidos com o mecanismo de escalonamento AISA são expostos e analisados. Foram criados três cenários de simulação. Para cada um, os parâmetros do AISA foram ajustados conforme a métrica de desempenho enfatizada. O primeiro cenário busca aumentar a vazão dos fluxos entre picorredes. O segundo

cenário tem como objetivo minimizar o retardo do tráfego que passa pelas pontes. O terceiro busca otimizar o desempenho das pontes quanto ao consumo de energia.

Em todas as simulações, o parâmetro α da fórmula de cálculo da utilização média do enlace entre a ponte e uma picorrede foi configurado para 0,6. Conseqüentemente, a fórmula resulta em $\mathbf{avg\_util} = \mathbf{0,6} \times \mathbf{avg\_util} + \mathbf{0,4} \times \mathbf{curr\_util}$. Também foram realizados alguns experimentos com $\alpha = 0,8$ e $\alpha = 0,4$. Com o primeiro valor, o algoritmo AISA demora a se adaptar às mudanças no tráfego, pois são necessários vários turnos até que um fluxo, partindo de uma utilização próxima de zero (por exemplo, após um período de silêncio), atinja o limite **inc_bound** e possa aumentar a duração do encontro. O segundo valor de α aumenta muito a influência da última medição frente à informação de utilização média, provocando mudanças bruscas neste parâmetro. Em alguns testes, chegou a ocorrer aumento da duração do encontro em um turno e diminuição do encontro no turno seguinte. Optou-se, então, por $\alpha = 0,6$.

Além disso, em todos as topologias, a distância entre o mestre e os escravos é sempre menor que sete metros, evitando, assim, a influência do modelo de erros provocados pela distância nas simulações.

Nos cenários um e três, foram realizadas comparações de consumo de energia. A seguir, explicam-se as fórmulas empregadas no cálculo deste consumo. Posteriormente, é realizada a descrição de cada cenário, seguida pelos resultados correspondentes.

5.4.1 Cálculo do Consumo de Energia

O objetivo principal do cenário 3 é minimizar o consumo de energia. Mesmo nos cenários que enfatizam outras métricas de desempenho, o AISA também busca reduzir o consumo de energia, sem prejuízo da métrica principal. Esta subseção explica como é calculada a energia consumida nas simulações.

Foram adotados três níveis de consumo de corrente [3] para as estações: o nível mais alto (chamado cTx), durante a transmissão (tx) ou recepção (rx) de dados, o nível

[3] As fichas técnicas trabalham com os valores de consumo de corrente. Como a voltagem é fixa, em torno de 3 volts, a comparação entre valores de corrente mantém a mesma proporcionalidade da comparação entre valores de potência.

intermediário (cAct), quando a estação está ativa, mas sem transmitir ou receber, e o nível mais baixo (cHold), durante o modo HOLD.

As fichas técnicas de módulos de rádio consultadas fornecem informações incompletas ([58], [59], [60], [61]). Por exemplo, algumas disponibilizam os valores de tx e rx, mas não implementam o modo de baixo consumo HOLD. Portanto, os valores foram estipulados sem muita preocupação em termos absolutos, mas mantendo a proporcionalidade entre os três níveis. Os valores adotados para cTx, cAct e cHold foram 50 mA, 20 mA e 0,1 mA, respectivamente.

A duração de um *slot* é 625 μs, porém, o período de tx/rx de um pacote de um *slot* é de 366 μs. No restante do tempo, a estação permanece no nível cAct. O mesmo ocorre para as transmissões de pacotes de três e cinco *slots*, onde os períodos de tx ou rx são, respectivamente, 1622 μs e 2870 μs. Todas as estações precisam receber o início dos pacotes para descobrir o seu destinatário, o que consome cerca de 80 μs de um *slot*. Para simplificar os cálculos, esse período é desconsiderado. Estas estimativas de tempo foram extraídas do artigo de Linsky [62].

O cálculo da corrente consumida na transmissão ou recepção de pacotes de um *slot*, chamada de c(tx1), é feito pela fórmula: $c(tx1) = (cTx \times 366 + cAct \times 259)/625$. Analogamente, definem-se as fórmulas: $c(tx3) = (cTx \times 1622 + cAct \times 253)/1875$ e $c(tx5) = (cTx \times 2870 + cAct \times 255)/3125$.

O valor adotado como unidade de energia (u.e.) foi a potência de tx/rx do pacote de um *slot*. Portanto, $pps(tx1) = 1u.e.$, onde pps é a abreviatura de potência consumida por *slot*. As fórmulas empregadas no cálculo do consumo por *slot*, relativas ao c(tx1) são:

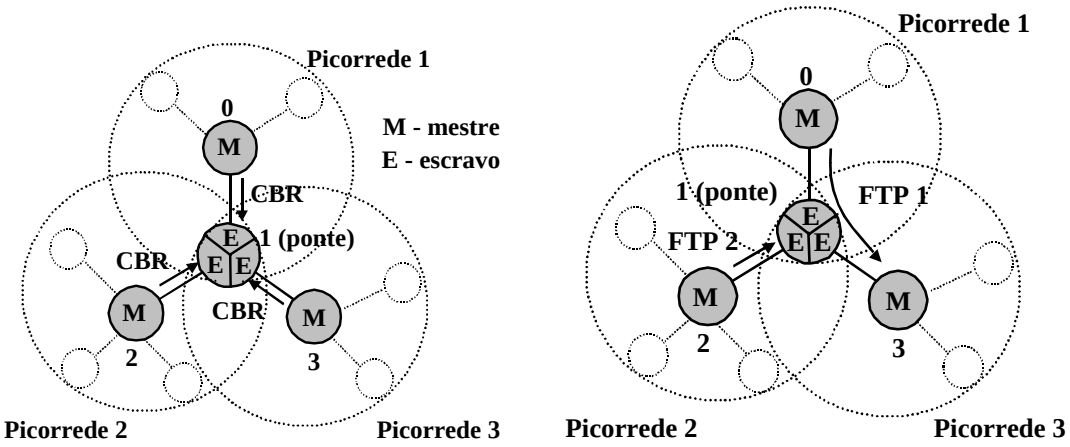
- $pps(tx1) = 1\ u.e.$;
- $pps(tx3) = c(tx3)/c(tx1) = 1,2232\ u.e.$;
- $pps(tx5) = c(tx5)/c(tx1) = 1,2658\ u.e.$;
- $pps(act) = cAct/c(tx1) = 0,5324\ u.e.$;
- $pps(hold) = cHold/c(tx1) = 0,0027\ u.e.$.

5.4.2 Cenário 1 - Métrica: Vazão

Neste cenário, os parâmetros do AISA são ajustados, buscando maximizar a vazão dos fluxos entre picorredes. Foi criada uma *scatternet*, composta por três picorredes, compartilhando uma estação ponte. A ponte comporta-se como escravo em todas as picorredes. O enfoque está voltado para a comunicação da ponte com os três mestres.

O cenário foi dividido em dois subcenários quanto ao tráfego na rede: o primeiro, apenas para ilustrar a influência da duração do turno na vazão e o segundo, contendo os demais resultados.

O primeiro subcenário foi configurado com três fontes de dados com taxa constante (CBR), uma em cada mestre, destinadas à ponte. Cada fonte gera dados à taxa de 500 kbps, em pacotes de 512 bytes, garantindo a plena ocupação do canal durante toda a simulação. O algoritmo usado pela ponte é o AISA. Este subcenário é ilustrado pela Figura 5.6 (a).



(a) Fontes CBR mantêm os enlaces sempre ocupados.

(b) Tráfego FTP/TCP não-persistente.

Figura 5.6: Configuração do cenário 1.

O segundo subcenário foi configurado com uma fonte de tráfego FTP no mestre da picorrede um, destinada ao mestre da picorrede três e uma fonte FTP no mestre da picorrede dois, destinada à ponte. Após a transmissão de cada arquivo, ambas as fontes FTP aguardam um intervalo, antes de iniciarem a transmissão do arquivo seguinte. O protocolo TCP New Reno empregado é não-persistente, pois a transmissão de cada

novo arquivo é iniciada com o TCP em *slow start*. Tanto o tamanho do arquivo quanto o intervalo entre o fim de um arquivo e o início do próximo seguem distribuições exponenciais. No tráfego entre as estações zero e três da Figura 5.6 (b), o tamanho médio do arquivo é de 30 Kbytes e o intervalo médio entre arquivos é de 1 segundo. Já no tráfego entre dois e um, estes valores são 40 Kbytes e 1 segundo, respectivamente.

O objetivo do modelo de tráfego utilizado neste subcenário é criar uma certa aleatoriedade nos fluxos de dados, alternando períodos de transmissão e silêncio. Os fluxos podem representar, por exemplo, a transferência de fotografias de uma câmera digital para um microcomputador portátil.

A Tabela 5.1 contém os valores dos parâmetros do AISA que foram mantidos fixos ao longo de todas simulações do primeiro cenário.

De um modo geral, os parâmetros foram escolhidos visando a rápida adaptação às mudanças no tráfego, maximizando a vazão média agregada. A duração mínima da ponte em uma picorrede (**min_dur**) deve ser a menor possível, liberando o máximo de *slots* para as picorredes com mais tráfego. Entretanto, deve ser longo o suficiente para permitir a transmissão de alguma informação e permitir a negociação da entrada da ponte em modo HOLD. Esta negociação pode ocupar até 10 *slots*, conforme explicado na Subseção 4.3.1. Portanto, configurou-se **min_dur** para 20 *slots*, garantindo-se 10 *slots* a mais para a transmissão de dados ou para a saída antecipada da ponte.

min_dur	max_dur	inc_rate	dec_rate	skip_pico	early_exit
20 *slots*	turn_sz *slots*	20%	20%	0 (bool)	turn_sz *slots*

Tabela 5.1: Parâmetros do AISA adotados no cenário 1.

A duração máxima da ponte em uma picorrede (**max_dur**) foi configurada com o valor da duração do turno (**turn_sz**), indicando que não há preocupação quanto à duração máxima da ponte em uma picorrede.

Quanto às taxas de acréscimo (**inc_rate**) e decréscimo (**dec_rate**) de *slots*, foram realizados testes com valores aditivos (5, 10 e 20 *slots*) e taxas multiplicativas (0,1, 0,2 e 0,4). Os resultados comparativos entre valores aditivos e multiplicativos não foram muito conclusivos e merecem um estudo mais detalhado [4]. É intuitivo que uma taxa de

[4]O TCP usa um modelo AIMD (*additive increase, multiplicative decrease*), agressivo, para o con-

incremento alta permita que um fluxo ganhe *slots* mais rapidamente, aumentando sua vazão. Contudo, o acréscimo de *slots* é dependente da quantidade de *slots* disponíveis, limitando, na maioria das vezes, o efeito da diferença entre os valores de **inc_rate**.

Finalmente, **skip_pico** e **early_exit** visam, exclusivamente, economizar energia e podem até degradar o desempenho da ponte em termos de vazão. Por exemplo, caso a ponte deixe uma picorrede antecipadamente por não haver dados a transmitir, no intervalo entre sua saída e o fim da duração prevista originalmente, pode chegar algum pacote, o qual não será transmitido. Por isso, **skip_pico** foi desabilitado e **early_exit** foi configurado com um valor alto, impossibilitando a saída antecipada da ponte.

5.4.3 Resultados do Cenário 1

O primeiro subcenário foi executado durante 180 segundos de simulação. As três fontes geram dados a taxas constantes e iguais. Variou-se o número de *slots* por turno e mediu-se a vazão agregada dos três fluxos em cada caso. Como não há aleatoriedade no cenário, não houve necessidade de executá-lo múltiplas vezes. O resultado desta simulação é mostrado na Figura 5.7.

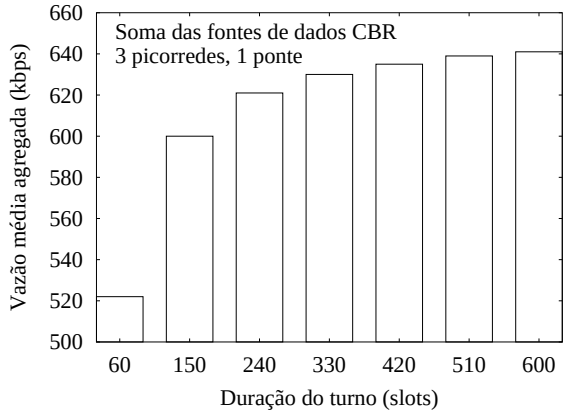


Figura 5.7: Vazão agregada média das fontes CBR, variando-se a duração do turno.

A ponte perde cerca de dois *slots* a cada troca de picorrede, para se sincronizar com a nova rede. Quanto menor a duração do turno, maior será a taxa de *slots* perdidos na troca de picorredes pela ponte. De 60 para 240 *slots*, a vazão média varia

trole de congestionamento. Entretanto, no caso do escalonamento, é possível que este modelo gere uma subutilização do canal, não devendo ser aplicado sem um estudo mais preciso.

bastante, conforme a Figura 5.7. Entretanto, a partir de 240 *slots* (equivalente a 80 *slots*/picorrede), esta variação é pequena. Por exemplo, a diferença entre o resultado para os turnos de 600 e 330 *slots* é de cerca de 10 kbps. Por outro lado, quanto maior for a duração do turno, maior será o retardo médio dos pacotes, já que a ponte permanece mais tempo ausente de cada picorrede. Este retardo pode crescer ainda mais no caso de um fluxo atravessando múltiplas pontes.

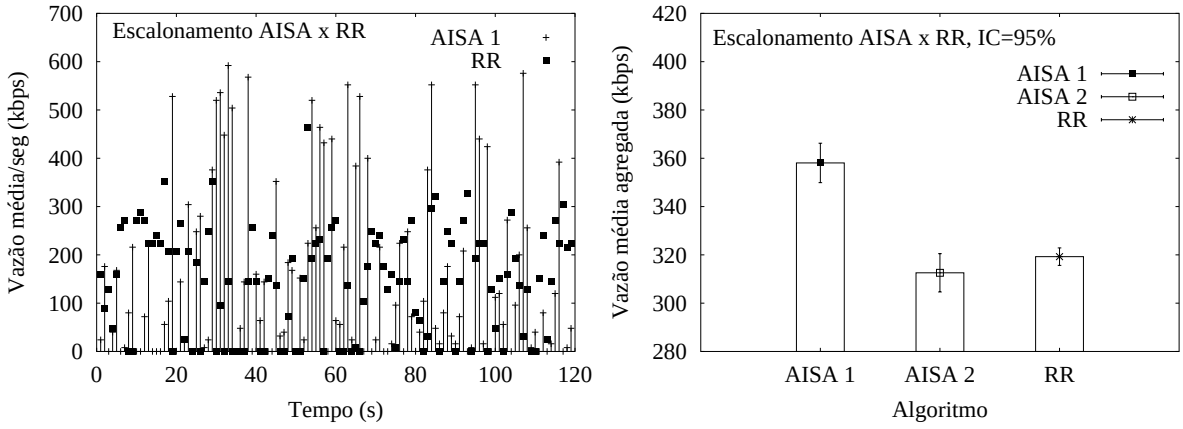
Nesta simulação, a escolha de um valor de turno entre 240 e 330 parece ser a mais sensata, garantindo entre 80 a 110 *slots*/picorrede no início da simulação. O efeito da troca de picorredes pela ponte é pequeno, e a vazão está próxima da maior obtida.

Um detalhe não mostrado pela Figura 5.7 é que, como a taxa de geração de pacotes das três fontes é alta, o AISA divide igualmente a duração do turno (**turn_sz**) entre as três picorredes, proporcionando o compartilhamento justo do enlace. Mesmo com as fontes iniciando em instantes diferentes, não ocorre o problema de captura do canal.

O segundo subcenário mostra o resultado comparativo entre o AISA e o *Round Robin* (RR). O AISA foi executado com duas configurações distintas. Na configuração AISA 1, os limites **inc_bound** e **dec_bound** foram definidos como 80% e 60%, respectivamente. Na configuração AISA 2, estes parâmetros assumiram, respectivamente, os valores 90% e 50%. Em todos os casos, a duração do turno foi configurada para 240 *slots*, propiciando, no início da simulação, 80 *slots*/picorrede. Foram executadas dez rodadas de simulação, cada uma com a duração de 120 segundos de duração. No cálculo das médias de todas as rodadas, empregou-se o intervalo de confiança de 95%.

O gráfico da Figura 5.8 (a) mostra a vazão agregada das duas fontes FTP, medida a cada segundo, para as configurações AISA 1 e *Round Robin* (RR). Este resultado representa apenas uma das dez rodadas, mas todas tiveram comportamente similar. A grande oscilação da vazão ao longo do tempo é provocada pela aleatoriedade do tráfego FTP/TCP não-persistente. O algoritmo AISA permite que o tráfego alcance vários picos de vazão acima de 500 kbps, representados na figura pelas barras verticais. Estes picos são causados pela redistribuição dinâmica de *slots* propiciada pelo AISA. Já no caso do algoritmo RR, apenas um ponto encontra-se acima de 400 kbps.

A Figura 5.8 (b) mostra a vazão média agregada dos fluxos FTP, computada apenas



(a) Vazão por segundo

(b) Média da simulação

Figura 5.8: Comparação da vazão com os mecanismos AISA e RR.

nos períodos de transmissão de dados. O resultado é a média das dez rodadas, com intervalo de confiança de 95%. A configuração AISA 1 apresentou a vazão média próxima de 360 kbps, contra os 320 kbps do RR, propiciando um ganho acima de 10% no desempenho.

O desempenho da configuração AISA 2 foi bem inferior ao da AISA 1. Conforme o limite **inc_bound** aproxima-se dos 100%, a utilização média **avg_util** demora mais a ultrapassá-lo, retardando o aumento da duração da ponte em uma picorrede (**next_dur**). Analogamente, quanto maior for o limite **dec_bound**, mais rapidamente uma picorrede pode liberar *slots*. A configuração AISA 2 torna a duração da ponte nas picorredes (**next_dur**) mais estável, diminuindo a adaptabilidade do algoritmo.

Analisando as duas fontes FTP separadamente, constata-se que quando apenas o tráfego da picorrede dois para a ponte está presente, este pode ocupar até 200 *slots* do turno ($\mathbf{turn\_sz} - \mathbf{2} \times \mathbf{min\_dur}$). Nos momentos em que o tráfego FTP da picorrede 1 para a 3 é o único existente, tanto a picorrede 1 quanto a 3 ocuparão 110 *slots* cada ($(\mathbf{turn\_sz} - \mathbf{min\_dur})/\mathbf{2}$). O ganho maior propiciado pelo AISA ocorre, portanto, quando apenas o tráfego destinado à ponte está presente.

Apesar de a economia de energia não ser o foco desta simulação, o consumo de energia foi medido no AISA 1 e no *Round Robin* (RR). Os valores apresentados na Tabela 5.2 são medidos em termos da unidade de energia (u.e.) definida na Seção 5.4.1

e representam a média das dez rodadas de simulação, com intervalo de confiança de 95%.

Valor medido em u.e.	AISA 1	RR
Energia total consumida	153944 ± 5516	206642 ± 433
Consumo em modo HOLD	146 ± 9	0
Consumo dos pacotes Poll/Null	43998 ± 734	106292 ± 1867

Tabela 5.2: Comparação do consumo de energia entre AISA e RR.

A tabela mostra que AISA consumiu cerca de 25% menos energia que o RR. Mesmo com os parâmetros **skip_pico** e **early_exit** desativados, a economia de energia é causada pela liberação de *slots* destinados às picorredes, nos períodos em que estas apresentam utilização média abaixo do limite **dec_bound**. Nestes períodos, terminado o escalonamento de todas as picorredes, a ponte permanece o restante do turno em modo HOLD.

Considerando que a vazão agregada do AISA foi superior à obtida com o RR, a relação u.e./byte transmitido é bem menor para o AISA. Mais da metade da energia consumida pela simulação com o RR é causada pela transmissão de pacotes Poll/Null, quando não há dados a transmitir. O valor médio de 146 u.e. gastos no modo HOLD equivale ao tempo de cerca de 55000 *slots*, ou seja, quase 30% da duração da simulação (120 s ≡ 192000 *slots*).

Com o auxílio dos resultados, fica claro que alguns parâmetros influem diretamente na vazão agregada da *scatternet*. As seguintes sugestões ajudam a aumentar a vazão. A duração do turno (**turn_sz**) deve ser grande, garantindo entre 80 e 100 *slots*/picorrede no início da simulação. A duração mínima da ponte em uma picorrede (**min_dur**) deve ser reduzida, ao passo que não se deve restringir sua duração máxima (**max_dur**). Valores próximos de 100% para o limite **inc_bound** dificultam o ganho de *slots*. Analogamente, valores próximos de zero para o limite **dec_bound** dificultam a liberação de *slots*. Com os valores de 100% e zero para **inc_bound** e **dec_bound**, o funcionamento do AISA torna-se similar ao *Round Robin*.

Os parâmetros **skip_pico** e **early_exit** ajudam a economizar energia quando há pouco tráfego de dados. Portanto, estes parâmetros têm pouca influência na vazão agregada, porque afetam justamente os tráfegos que já têm uma vazão baixa.

5.4.4 Cenário 2 - Métrica: Retardo

Este cenário é composto por duas picorredes. A primeira contém cinco dispositivos. Um *laptop* funciona como mestre, e a este estão conectados outro *laptop*, um *mouse* sem fio e dois PDAs. Na segunda picorrede, existe um microcomputador (mestre) ligado a uma impressora sem fio. Em certo momento, o *laptop* conecta-se ao microcomputador, para imprimir um arquivo. Nesta segunda picorrede, o *laptop* passa a funcionar como escravo, caracterizando uma ponte mestre/escravo. A Figura 5.9 ilustra este cenário.

O *mouse* é um dispositivo que gera pouco tráfego, mas que exige uma certa interatividade, situando-se na categoria dos dispositivos de tráfego com requisito de retardo limitado. O *mouse* gera um pacote de 16 bytes a cada 65 ms (valor próximo do adotado por Racz et al. [14]), caracterizando uma taxa de transmissão de 2,56kbps.

O tráfego de fundo é composto pelo fluxo de impressão e três fluxos na primeira picorrede. A impressão foi modelada como um FTP/TCP New Reno. Os outros fluxos são um FTP/TCP New Reno, um CBR com taxa de transmissão de 50kbps e pacotes de 512bytes e um CBR com taxa de transmissão de 200kbps e pacotes de 512bytes. Estes três fluxos de fundo na primeira picorrede visam apenas aumentar a ocupação do enlace.

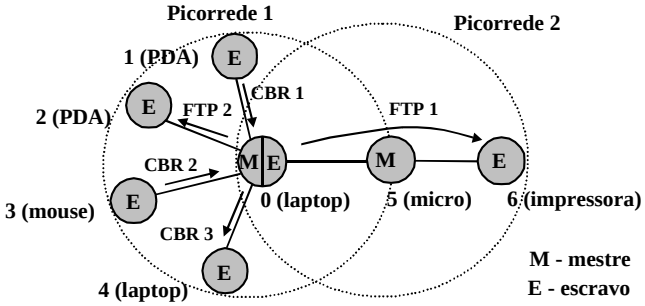


Figura 5.9: Configuração do cenário 2. No subcenário 2.1, há apenas o FTP 1 e o CBR 2. No subcenário 2.2, todo o tráfego está presente.

O comportamento do *mouse* será avaliado em dois subcenários. No Subcenário 2.1, existe apenas tráfego interpicorrede funcionando como tráfego de fundo. No Subcenário 2.2, o tráfego interno à picorrede 1 é incluído na simulação. Neste último caso, os seguintes algoritmos intrapicorrede serão testados na picorrede 1: *Round Robin* (RR), *Deficit Round Robin* (DRR) e *Deficit Round Robin with Quality of Service* (DRR-CoS).

Tanto no DRR quanto no DRR-CoS, o parâmetro *quantum* é igual a 15. Um *quantum* menor tornaria o comportamento do DRR parecido com o do RR, conforme explicado na Subseção 5.3.1. Por exemplo, se o *quantum* fosse cinco e o tamanho dos pacotes do tráfego fosse cinco *slots*, o DRR propiciaria os mesmos resultados que o RR. O parâmetro **IPol** do DRR-CoS está configurado com o valor de 60 ms, visando garantir retardos menores que 65 ms para os pacotes do *mouse*.

A Tabela 5.3 mostra os valores dos parâmetros do AISA que foram mantidos fixos ao longo de todas simulações do cenário 2.

inc_bound	dec_bound	inc_rate	dec_rate	max_dur	skip_pico	early_exit
80%	60%	20%	20%	turn_sz *slots*	0 (bool)	turn_sz *slots*

Tabela 5.3: Parâmetros do AISA adotados no cenário 2.

Como a taxa de geração de pacotes do tráfego do *mouse* é baixa, os parâmetros que permitem ganhar ou liberar *slots* mais rapidamente (**inc_bound**, **dec_bound**, **inc_rate** e **dec_rate**) têm pouca influência no seu desempenho, mas podem, sem prejuízo ao *mouse*, melhorar o desempenho do tráfego de fundo. Por isso, foram adotados os mesmos valores testados no Cenário 1.

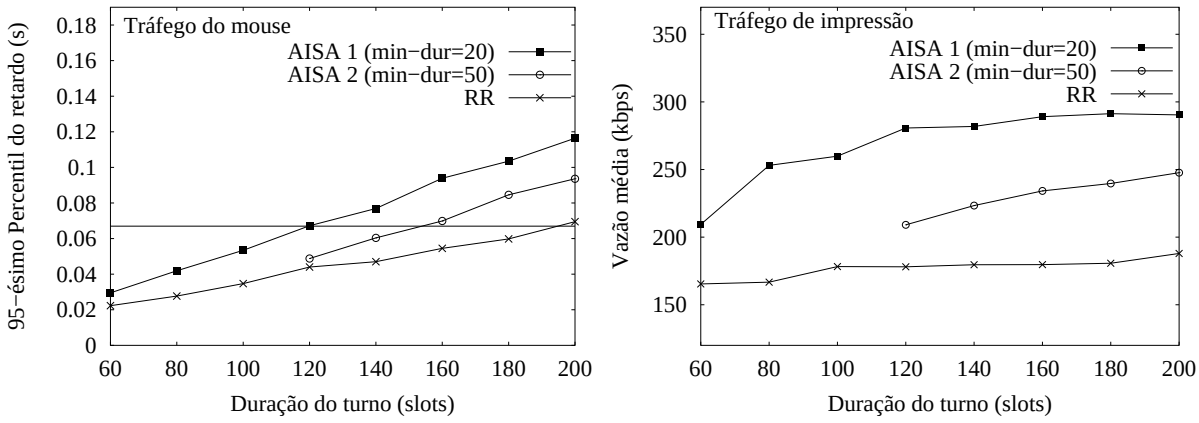
Novamente, **skip_pico** e **early_exit** foram configurados de modo a desativar suas funcionalidades de economia de energia. O motivo é o mesmo do cenário 1: evitar que um pacote do tráfego do *mouse* deixe de ser transmitido, por estar a ponte, naquele instante, em modo de baixo consumo.

5.4.5 Resultados do Cenário 2

Inicialmente, foi simulado o Subcenário 2.1, com a presença apenas do tráfego gerado pelo *mouse* e do tráfego de impressão. As simulações tiveram a duração de 120 segundos e foram comparadas três configurações diferentes de algoritmo de escalonamento interpicorrede na ponte: AISA 1, com a duração mínima da ponte em uma picorrede (**min_dur**) igual a 20 *slots*, AISA 2, com **min_dur** igual a 50 *slots* e *Round Robin* (RR). Para cada configuração, variou-se a duração do turno de 60 a 200 *slots*, exceto na segunda configuração, onde as simulações foram iniciadas em 120 *slots*.

A Figura 5.10 (a) mostra o 95-ésimo percentil do retardo para as três configurações

acima, para as diversas durações de turno (**turn_sz**). Como o tráfego do *mouse* foi modelado gerando um pacote a cada 65 ms, considerou-se este o valor máximo aceitável para o retardo dos pacotes.



(a) Comparação do retardo

(b) Comparação da vazão média

Figura 5.10: Resultados comparativos entre AISA e RR, apenas com impressão de fundo.

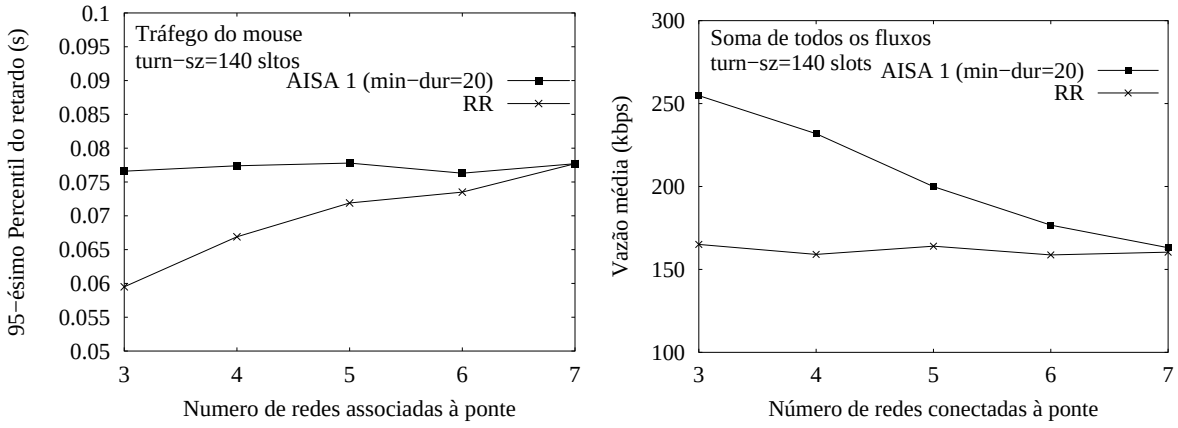
A configuração AISA 1 ultrapassa o limite estipulado próximo dos 120 *slots*. Já com o aumento da duração mínima (configuração AISA 2), o limite de retardo só é ultrapassado próximo de 150 *slots*. Este fato ocorre porque aumentando-se **min_dur**, diminui-se automaticamente a duração máxima da ponte na outra picorrede.

Como a vazão do *mouse* é baixa, a ponte dedica o tempo mínimo (**min_dur**) a sua picorrede. Quando **min_dur** é 20 *slots* e o turno tem duração (**turn_sz**) de 120 *slots*, 100 *slots* (62,5 ms) destinam-se à picorrede com o tráfego de impressão. Por isso, este valor está próximo do limite superior do retardo aceitável para o *mouse*.

O algoritmo *Round Robin* (RR) fornece retardos menores que o AISA para uma mesma duração de turno. Os valores são, inclusive, bem inferiores ao limite de 65 ms estipulado. Entretanto, esta redução do retardo vem acompanhada de uma diminuição na vazão agregada do tráfego de fundo. Este efeito é representado na Figura 5.10 (b). A vazão média do tráfego de impressão usando o RR fica abaixo de 180 kbps para todos os valores de turno testados. Por sua vez, o AISA 1 obtém uma vazão média em torno de 280 kbps para o tráfego de impressão, mantendo o retardo dentro do limite estabelecido.

O Subcenário 2.1 contém apenas duas picorredes. Entretanto, convém analisar o comportamento do retardo dos pacotes do *mouse* e da vazão do tráfego de impressão, quando a ponte participa de mais de duas picorredes. Variou-se o número de picorredes conectadas à ponte de três até sete. O teste foi executado com turno de 140 *slots*. Este valor foi escolhido para permitir que, mesmo na configuração com sete picorredes, a duração mínima (**min_dur**) de 20 *slots* seja respeitada para todas as redes. Além disso, com sete picorredes, o algoritmo AISA deve se comportar como o RR, pois todas as picorredes permanecem com a mesma parcela do turno.

Cada picorrede adicionada ao cenário mantém um fluxo de transferências de arquivo (FTP) sobre TCP New Reno não-persistente, análogo ao explicado no cenário anterior. Os arquivos transferidos têm tamanho médio de 10 Kbytes e o intervalo médio de silêncio entre arquivos é de 1 s. Os resultados do 95-ésimo percentil do retardo dos pacotes do *mouse* e a vazão agregada de todo o tráfego são mostrados na Figura 5.11.



(a) Comparação do retardo

(b) Comparação da vazão média

Figura 5.11: Resultados comparativos entre AISA e RR, variando-se o número de picorredes conectadas à ponte.

O perfil do retardo para o AISA 1 se mantém estável, independente do número de picorredes, pois a picorrede associada ao *mouse* sempre ocupa a duração mínima permitida. A ponte divide o tempo restante entre as outras picorredes. Já o *Round Robin* (RR) reparte o turno igualmente entre todas as redes. Com o RR, a picorrede do *mouse* inicia com a metade do turno de presença da ponte (para duas picorredes) e termina com um sétimo do turno (para sete picorredes). Por isso, o retardo cresce com

o aumento do número de picorredes associadas à ponte. Apesar do AISA manter os valores de retardo maiores do que o RR, o AISA consegue, através da parametrização, obter retardos bem definidos em todos casos.

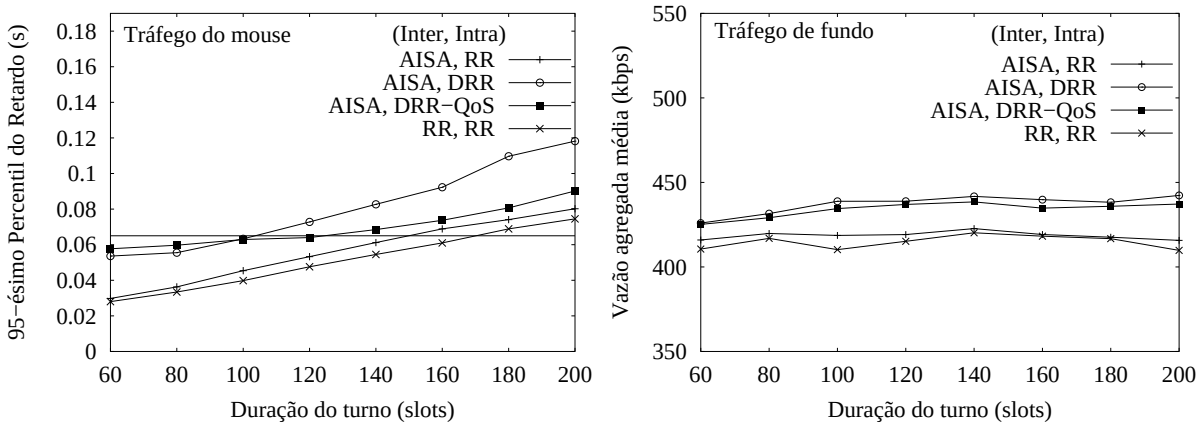
Em termos de vazão, o RR mantém a curva estável, devido a dois fatores de tendências opostas. Por um lado, a vazão deveria aumentar com o número de picorredes, pois a permanência da ponte na picorrede do *mouse*, com pouco tráfego, reduz. Por outro lado, o crescimento do número de picorredes aumenta a quantidade de trocas entre estas pela ponte, provocando o desperdício de *slots*.

A vazão agregada obtida com o AISA diminui com o aumento do número de picorredes. Com duas picorredes, a ponte permanece a maior parte do turno voltada para a atividade de impressão. Conforme são acrescentadas novas redes, diminui-se esta permanência. Além disso, perdem-se cerca de dois *slots* a cada troca de picorredes realizada pela ponte. Ainda assim, na configuração com seis picorredes, a vazão agregada com o AISA é cerca de 10% maior do que com o algoritmo RR.

O Subcenário 2.2 apresenta outra configuração de tráfego de fundo. Três fontes de dados de melhor esforço, localizadas em estações da picorrede do *mouse*, foram adicionadas ao cenário. Três algoritmos de escalonamento intrapicorrede foram empregados nesta picorrede: *Round Robin* (RR), *Deficit Round Robin* (DRR) e *Deficit Round Robin with Quality of Service* (DRR-CoS). A simulação teve a mesma duração daquela realizada na primeira parte deste cenário. O AISA foi configurado com a duração mínima (**min_dur**) igual a 20 *slots*.

A Figura 5.12 (a) mostra o 95-ésimo percentil do retardo para os diversas durações de turno (**turn_sz**), com as quatro configurações testadas. Como existem vários fluxos de dados destinados à ponte na picorrede 1, a ponte passa a dividir igualmente sua presença nas duas picorredes. Portanto, o tráfego de impressão deixa de ser o maior causador de retardo para os pacotes do *mouse*.

A grande quantidade de tráfego na picorrede 1 faz com que o algoritmo de escalonamento DRR apresente a curva com os maiores valores de retardo para o *mouse*. Já as curvas (RR, RR) e (AISA, RR) são bem parecidas, pois a existência de muito tráfego entre a ponte e ambas as picorredes torna o desempenho do escalonamento



(a) Comparação do retardo

(b) Comparação da vazão média

Figura 5.12: Resultados comparativos entre AISA e RR, onde o tráfego do *mouse* compartilha o enlace com várias fontes de tráfego de fundo.

interpicorrede AISA similar ao RR.

O DRR-CoS tem a preocupação de manter o retardo do tráfego do *mouse* limitado. Por isso, este algoritmo apresenta a curva de menor inclinação. O limite de 65 ms é atingido com a duração do turno em 120 *slots*. Este resultado é o mesmo obtido no subcenário 2.1, empregando-se o AISA 1. Portanto, a utilização conjunta do AISA, para o escalonamento interpicorrede, com o DRR-CoS, para o escalonamento intrapicorrede, parece ser a mais indicada para manter o retardo de um tráfego limitado.

Em termos de vazão, a Figura 5.12 (b) mostra que as quatro configurações obtiveram resultados próximos, com uma ligeira vantagem para as configurações, cujo algoritmo intrapicorrede é baseado no *Deficit Round Robin* (DRR). Os resultados são parecidos porque a carga na rede é alta, havendo, quase sempre, pacotes a serem transmitidos. A pequena vantagem do DRR é ocasionada pela configuração do *quantum*. O DRR só troca de estação quando o *quantum* é zero. Já o *Round Robin* (RR) escalona uma estação por vez, passando mais vezes pelo *mouse* em cada turno.

Dos resultados expostos, conclui-se que o AISA pode ser configurado para manter o retardo limitado para determinados fluxos, aumentando, concomitantemente, a vazão dos fluxos restantes. Sugere-se a configuração da duração do turno de forma que a ponte não permaneça ausente de uma picorrede por mais tempo do que o estipulado para o

retardo. Por exemplo, para o retardo máximo de 65 ms (equivalente a 104 *slots*), pode-se usar uma configuração com a duração do turno (**turn_sz**) e a permanência mínima da ponte em uma picorrede (**min_dur**) iguais a 120 e 20 *slots*, respectivamente, ou 140 e 40 *slots*, respectivamente. Em ambos os casos, se a ponte permanecer o mínimo de tempo em uma picorrede (**min_dur**), o período de sua ausência será de 100 *slots*.

5.4.6 Cenário 3 - Métrica: Consumo de Energia

O objetivo do Cenário 3 é mostrar que, configurando-se corretamente os parâmetros do AISA, pode-se reduzir o consumo de energia nas pontes e, conseqüentemente, na *scatternet* como um todo. Dependendo da configuração, o AISA pode se tornar mais eficaz na economia de energia, mesmo que isso prejudique outras métricas de desempenho.

O Cenário 3 pode representar, por exemplo, uma pequena rede de sensores [63, 64]. Uma grande preocupação das redes de sensores é o consumo de energia e, dependendo da extensão da rede, Bluetooth pode ser uma alternativa de tecnologia de transmissão. A topologia do cenário é uma *scatternet*, composta por nove picorredes. Somente as picorredes que se encontram nas extremidades possuem fontes de dados. As fontes são divididas em três categorias, e cada picorrede possui uma fonte de cada categoria, totalizando três fontes por picorrede. A Figura 5.13 ilustra este cenário.

A estação central na *scatternet* é um ponto de acesso ao qual se destina todo o tráfego de dados. As três categorias de tráfego foram modeladas por fontes gerando dados à taxa constante (CBR). Todas as fontes geram pacotes à taxa de 3 kbps. Entretanto, para aumentar a heterogeneidade do tráfego, as categorias possuem tamanhos de pacote diferentes. A categoria 1 transmite pacotes de 300 bytes, a categoria 2, de 100 bytes e a categoria 3, de 20 bytes.

Neste cenário, apenas os parâmetros contidos na Tabela 5.4 foram mantidos fixos em todas as simulações.

A duração mínima da ponte em uma picorrede (**min_dur**) deve ser pequena, permitindo que a ponte permaneça pouco tempo em picorredes com poucos dados a transmitir. O parâmetro **skip_pico** permite que a ponte deixe de escalonar uma determinada

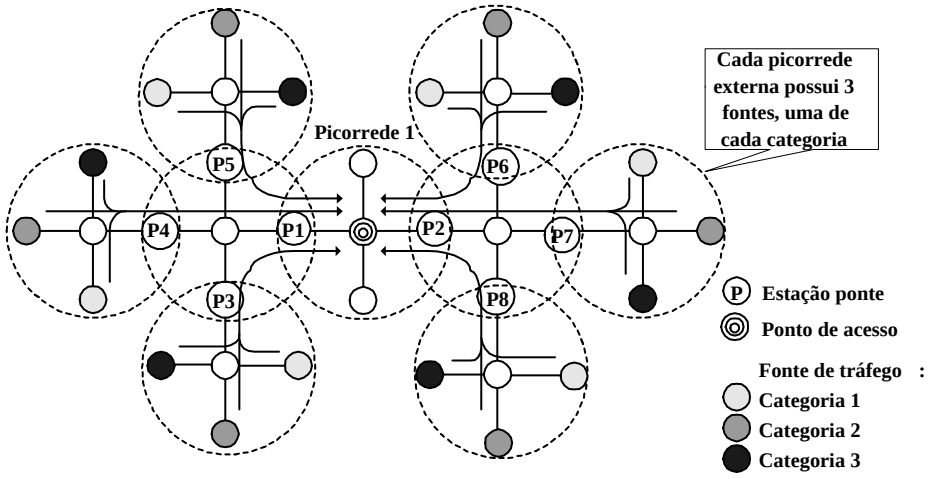


Figura 5.13: Topologia do cenário 3, voltado para a economia de energia.

picorrede em um turno, permanecendo a ponte, no período destinado a esta picorrede, em modo de baixo consumo. O valor de **early_exit** também possibilita a economia de energia.

min_dur	dec_rate	skip_pico	early_exit
20 *slots*	20%	1 (bool)	15 *slots*

Tabela 5.4: Parâmetros do AISA adotados no cenário 3.

5.4.7 Resultados do Cenário 3

O cenário foi testado com duas configurações do AISA e com o *Round Robin* (RR). O algoritmo RR, apesar de não propiciar economia de energia, serve como parâmetro de comparação em termos de retardo dos pacotes. A métrica de retardo não é o foco principal do cenário 3, mas a economia de energia não pode trazer como conseqüência um retardo excessivo dos pacotes. Dependendo da aplicação, um valor de retardo muito alto pode inutilizar a informação transmitida.

Os parâmetros das duas configurações do AISA, denominadas AISA 1 e AISA 2, são expostos na Tabela 5.5. As duas configurações pretendem identificar os parâmetros de maior influência no consumo de energia e, também, no retardo dos pacotes [5]. Em princípio, AISA 2 deve permitir maior economia de energia do que AISA 1, pois tem maior dificuldade em aumentar a duração do turno (maior **inc_bound**) e maior facilida-

[5]Foram testadas outras parametrizações do AISA. A comparação entre as duas selecionadas é suficiente para mostrar a influência dos parâmetros.

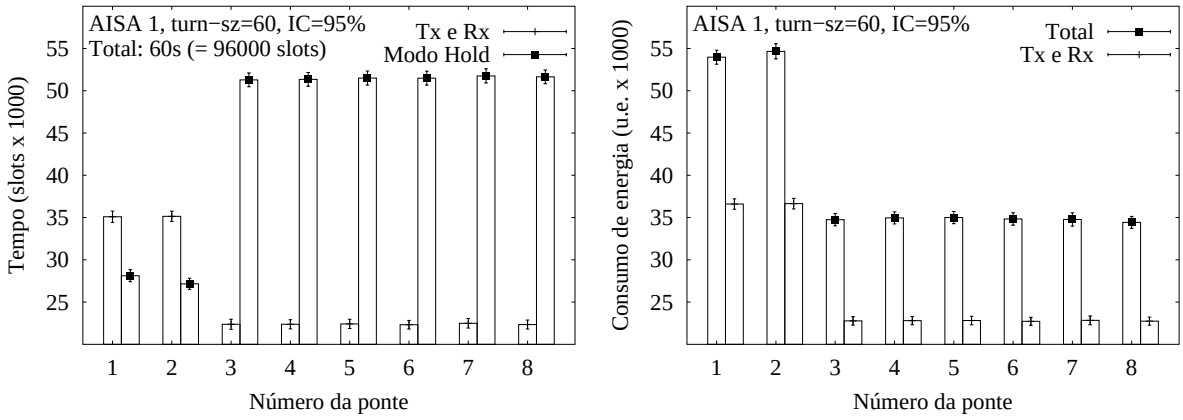
de em liberar *slots* (maior **dec_bound**). AISA 2 também ganha **slots** mais lentamente (menor **inc_rate**) e limita superiormente a permanência da ponte em uma picorrede (**max_dur**). Os efeitos destas configurações são analisados adiante nesta subseção.

	inc_bound	dec_bound	inc_rate	max_dur
AISA 1	80%	60%	20%	turn_sz *slots*
AISA 2	90%	70%	10%	0,6 × turn_sz *slots*

Tabela 5.5: Configurações AISA 1 e AISA 2.

Em uma primeira etapa, as simulações tiveram por objetivo medir o consumo de energia das pontes nas configurações do AISA e do RR. Foram executadas dez rodadas de simulações, com a duração de 60 segundos cada. Em cada uma das dez rodadas, variou-se a duração do turno de 60 até 140 *slots*.

O cenário contém oito pontes, numeradas segundo a Figura 5.13. Para cada ponte, foi tirada a média do consumo de energia das dez rodadas, com intervalo de confiança de 95%. Os resultados do consumo das pontes, com a duração do turno de 60 *slots*, para a configuração AISA 1 são apresentados na Figura 5.14 (a) e (b). O item (a) mostra os resultados em termos de *slots* e o item (b) os apresenta convertidos em unidades de energia (u.e.).



(a) Ocupação do tempo pelas pontes

(b) Consumo de energia

Figura 5.14: Consumo de energia das pontes na configuração AISA 1, com **turn_sz** igual a 60 *slots*.

A Figura 5.14 mostra que as pontes podem ser divididas em dois grupos, em relação à sua posição na topologia: o grupo 1, contendo as pontes 1 e 2 e o grupo 2, contendo as demais pontes. As pontes 1 e 2 estão a um salto de distância do ponto de acesso e cada uma recebe os dados coletados por três picorredes. As outras seis pontes também

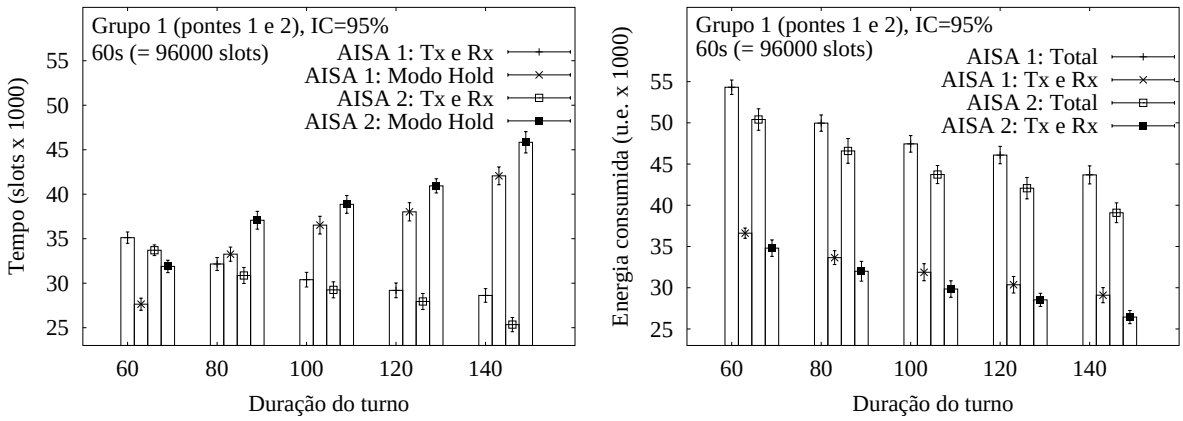
são equidistantes, em número de saltos, do ponto de acesso e encaminham a este, aproximadamente, a mesma quantidade de informação.

Como esperado, os resultados mostram que as pontes do grupo 1 recebem e transmitem muito mais dados (cerca de 35000 *slots* contendo dados) do que as pontes do grupo 2 (cerca de 22000 *slots*) e, por isso, permanecem menos tempo em modo HOLD (27000 *slots* do grupo 1 contra 51000 *slots* do grupo 2). Esta diferença se reflete no consumo de energia. As pontes 1 e 2 consomem em torno de 54000 u.e. cada em toda a simulação, enquanto as outras consomem entre 34000 e 35000 u.e. cada. Certamente a energia do grupo 1 acabará antes da energia do grupo 2. Este efeito será analisado adiante, ainda nesta seção.

Os resultados de consumo de energia das pontes 1 e 2, pertencentes ao grupo 1, foram aproximadamente iguais entre si. O mesmo foi constatado para as demais pontes, pertencentes ao grupo 2. Por isso, daqui em diante, os resultados das pontes 1 e 2 são representados pela média obtida para o grupo 1. Analogamente, os resultados das demais pontes são representados pela média obtida para o grupo 2. As Figuras 5.15 (a) e 5.16 (a) mostram o tempo ocupado com transmissão e recepção de pacotes e o tempo acumulado de permanência em modo HOLD para os dois grupos, com as configurações AISA 1 e AISA 2. Já as Figuras 5.15 (b) e 5.16 (b) apresentam a energia consumida pelos dois grupos na simulação [6].

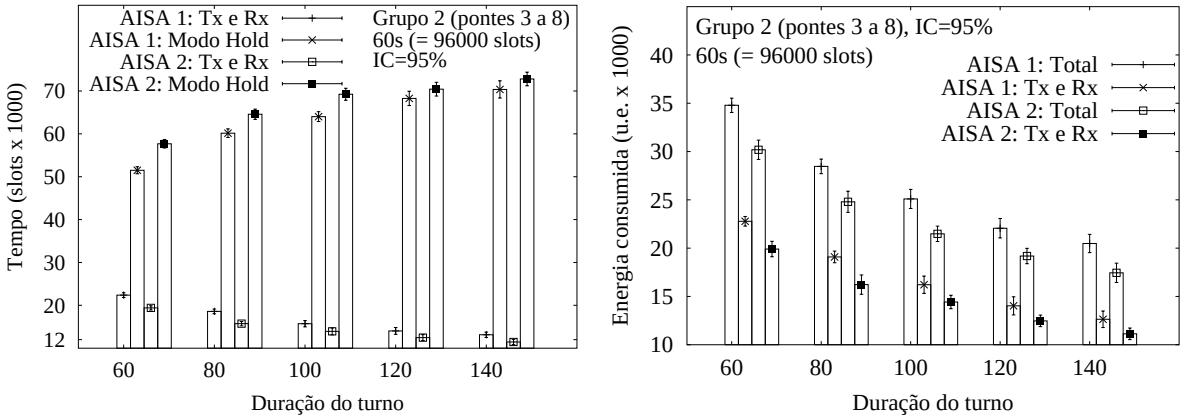
Em todos os casos, o aumento do turno provoca a redução da quantidade de pacotes transmitidos e recebidos pela ponte. Esta redução ocorre porque, como as pontes têm poucos dados a transmitir, estas permanecem, normalmente, o tempo mínimo (**min_dur**) em cada picorrede. O aumento da duração do turno possibilita que a ponte, após escalonar todas as suas picorredes, fique um período maior em modo HOLD, até o fim do turno. Conseqüentemente, há o aumento do período de permanência das pontes em modo HOLD, destacado nas Figuras 5.15 (a) e 5.16 (a). Esta situação se reflete na diminuição do consumo de energia com o aumento da duração do turno (Figuras 5.15 (b) e 5.16 (b)).

[6]Os resultados do RR não foram colocados na figura por motivo de legibilidade. O RR não economiza energia, por não trabalhar com o modo HOLD. Por exemplo, para o grupo 1, com turno de 60 *slots*, o consumo médio de energia com o RR foi 70461 $\pm$ 1023 u.e., ou seja, cerca de 30% maior do que usando o AISA 1.



(a) Ocupação do tempo pelo do grupo 1 (b) Consumo de energia do grupo 1

Figura 5.15: Consumo de energia das pontes do grupo 1, com as configurações AISA 1 e AISA 2, variando-se a duração do turno.



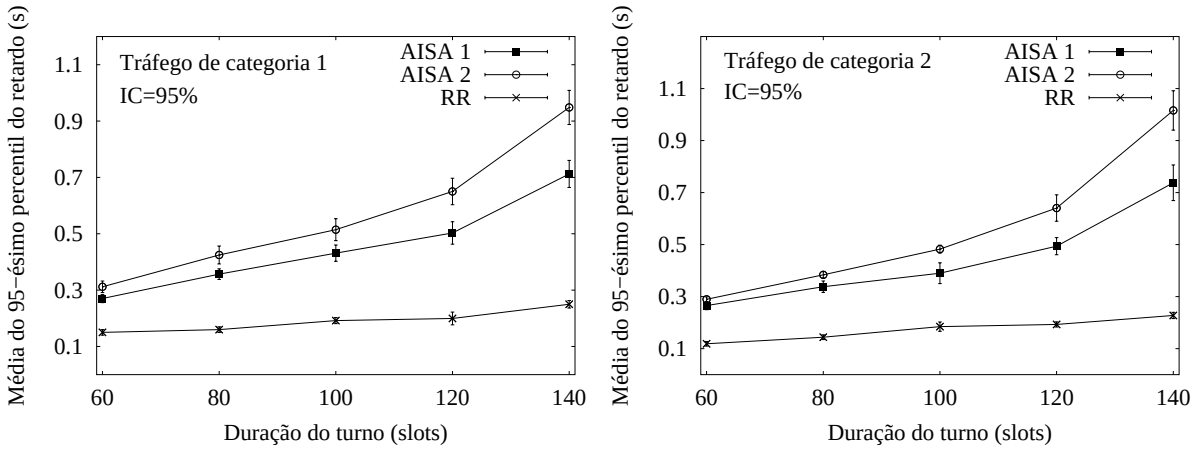
(a) Ocupação do tempo pelo do grupo 2 (b) Consumo de energia do grupo 2

Figura 5.16: Consumo de energia das pontes do grupo 2, com as configurações AISA 1 e AISA 2, variando-se a duração do turno.

Comparando as configurações AISA 1 e AISA 2 (ver Tabela 5.5), nota-se que AISA 2 propicia maior economia de energia do que AISA 1. Vários fatores contribuem para este efeito. Na configuração AISA 2, a ponte tem mais facilidade para liberar *slots* de uma picorrede (pois $\mathbf{dec\_bound_{AISA2}} > \mathbf{dec\_bound_{AISA1}}$) e mais dificuldade de aumentar sua duração em uma picorrede ($\mathbf{inc\_bound_{AISA2}} > \mathbf{inc\_bound_{AISA1}}$) do que na AISA 1. Quando a utilização média do enlace (**avg_util**) ultrapassa o limite que permite o ganho de *slots* (**inc_bound**), a taxa de acréscimo na configuração AISA 2 é

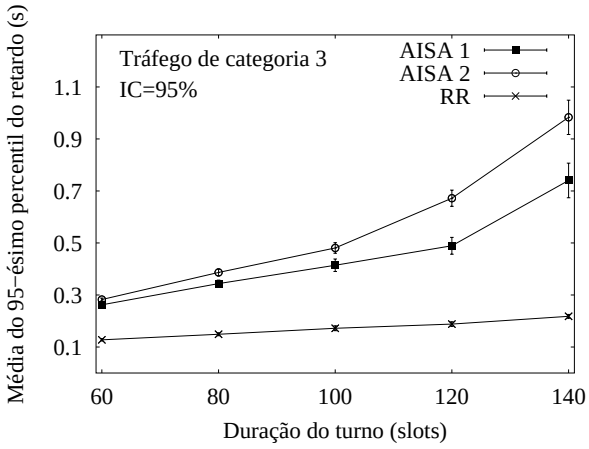
menor do que na AISA 1 ($\mathbf{inc\_rate_{AISA1}} < \mathbf{inc\_rate_{AISA2}}$). Finalmente, em AISA 1, a ponte pode ocupar todos os *slots* livres no encontro com uma picorrede, enquanto em AISA 2, a duração de um encontro é limitada superiormente por **max_dur**.

A maior economia de energia propiciada pela configuração AISA 2 traz como contrapartida o aumento do retardo dos pacotes. No estudo do retardo, incluiu-se o algoritmo *Round Robin* (RR) como parâmetro de comparação. O cenário contém 18 fontes de dados, sendo seis de cada categoria. Para cada rodada, o 95-ésimo percentil do retardo dos pacotes das 18 fontes foi tirado. Ao final das dez rodadas, foi calculada a média do 95-ésimo percentil do retardo para as fontes, com intervalo de confiança de 95%. A Figura 5.17 apresenta o resultado médio do retardo por categoria.



(a) Tráfego de categoria 1

(b) Tráfego de categoria 2



(c) Tráfego de categoria 3

Figura 5.17: Retardo dos pacotes das três categorias de sensores, com as configurações AISA 1, AISA 2 e RR.

As curvas de retardo são bastante parecidas nas três categorias. O algoritmo *Round Robin* (RR) propicia os menores retardos, porém, sem economia de energia. A configuração AISA 2 apresenta retardos ligeiramente maiores que AISA 1 até o valor de turno de 100 *slots*. A partir daí, a diferença entre as duas configurações aumenta em proporções maiores. Portanto, existe um compromisso entre a economia de energia desejada e o retardo aceitável para o tráfego.

Devido ao posicionamento das pontes na topologia, as pontes 1 e 2, pertencentes ao grupo 1, consomem mais energia do que as pertencentes ao grupo 2, conforme explicado anteriormente nesta subseção. Supondo que todas as pontes comecem a funcionar com a mesma carga de bateria, as pontes 1 e 2 deixarão de funcionar antes das restantes. Quando isso ocorrer, não haverá mais rota para o ponto de acesso. Portanto, em alguns casos, devem-se buscar configurações diferenciadas para as pontes, na tentativa de que todas esgotem sua energia ao mesmo tempo. A nova configuração não deve aumentar excessivamente o retardo dos pacotes.

Buscando equalizar o consumo de energia, o cenário 3 foi testado com duas parametrizações diferentes para o AISA, uma para cada grupo de pontes. A Tabela 5.6 resume as duas configurações. A parametrização do grupo 1 é similar à AISA 1 analisada anteriormente (ver Tabela 5.5), propiciando menor consumo de energia, porém, com retardos maiores. Já a parametrização do grupo 2 é análoga à AISA 2, com a qual se obteve o maior consumo de energia, mas com retardos menores.

	turn_sz	inc_bound	dec_bound	inc_rate	max_dur
Grupo 1	140 *slots*	90%	70%	10%	80 *slots*
Grupo 2	60 *slots*	80%	60%	20%	turn_sz *slots*

Tabela 5.6: Parametrização do AISA, visando equalizar o consumo de energia.

Esta simulação foi executada nas mesmas condições da simulação anterior, para permitir a comparação entre seus resultados. Foram calculados o consumo médio de energia para os dois grupos de sensores e a média do 95-ésimo percentil do retardo dos pacotes, ambos com intervalo de confiança de 95%. A Tabela 5.7 contém o resultado relativo ao consumo de energia nesta simulação, enquanto a Tabela 5.8 mostra os valores de retardo resultantes para as três categorias de tráfego.

A diferença entre a energia consumida pelas pontes dos grupos 1 e 2 foi menor que 3%. O mais importante é que o consumo das pontes 1 e 2 foi cerca de 10% menor do

	Grupo 1	Grupo 2
Dados transmitidos e recebidos (*slots*)	21989 ± 593	22451 ± 560
Tempo em HOLD (*slots*)	51756 ± 1017	52321 ± 985
Energia total consumida (u.e.)	35134 ± 775	34436 ± 591
Energia consumida na transmissão e recepção de pacotes (u.e.)	23136 ± 534	22952 ± 467

Tabela 5.7: Energia consumida, com configurações diferentes para os grupos 1 e 2.

	Categoria 1	Categoria 2	Categoria 3
Média do retardo	0,627 ± 0,015	0,57 ± 0,032	0,58 ± 0,024

Tabela 5.8: Retardo dos pacotes, com configurações diferentes para os grupos 1 e 2.

que o menor valor apresentado pelo AISA 2 na Figura 5.15 (b). O consumo foi, ainda, cerca de 20% menor do que a configuração AISA 2, com o turno de 120 *slots*, que apresenta os mesmos resultados de retardo (o retardo é avaliado adiante). Portanto, o estudo mostra que é possível obter uma configuração para as pontes que prolongue o tempo de funcionamento de toda a rede.

As pontes do grupo 1 foram configuradas de forma a economizar mais energia do que as outras pontes, tendo em vista seu maior tráfego de pacotes. Em contrapartida, o retardo dos pacotes que passam por estas pontes aumenta. Na tentativa de não aumentar ainda mais o retardo, as pontes do grupo 2 foram configuradas com parâmetros que propiciam menos economia de energia, mas mantêm o retardo o menor possível. Os valores de retardo (Tabela 5.8) ficaram na mesma faixa daqueles obtidos com a configuração AISA 2 e turno de 120 *slots*, mostrados na Figura 5.17.

Os resultados obtidos ao longo desta subseção mostram que o mecanismo AISA é uma boa alternativa para o escalonamento interpicorrede, quando o foco é a economia de energia. Além disso, mostrou-se ser possível empregar o AISA com parametrizações diferentes para as diversas pontes, prolongando a duração da rede.

Como regra geral, para se maximizar a economia de energia, devem-se aumentar a duração do turno (**turn_sz**) e os limites **inc_bound** e **dec_bound**. Deve-se limitar a duração máxima da ponte em uma picorrede (**max_dur**), e a taxa de acréscimo de *slots* (**inc_rate**) deve ser baixa. O efeito negativo desta configuração é o aumento do retardo dos pacotes.

5.5 Resumo

Este capítulo descreveu, inicialmente, a ferramenta de simulação BlueNetS, desenvolvida ao longo do trabalho de pesquisa para permitir o estudo de desempenho da tecnologia Bluetooth. As seções seguintes apresentaram os resultados das simulações dos algoritmos de escalonamento intrapicorrede (DRR-CoS) e interpicorrede (AISA) propostos.

Os resultados obtidos com o algoritmo DRR-CoS mostraram a viabilidade de se compartilhar o enlace sem conexão (ACL) entre o tráfego de melhor esforço e o tráfego com requisito de retardo limitado, maximizando a ocupação do canal, sem ultrapassar o referido limite de retardo.

Parâmetro	Descrição	Métrica de desempenho		
		Vazão	**Retardo**	**Consumo**
turn_sz	duração do turno	↑*	↓*	↑*
inc_bound	acima deste limite, o encontro pode aumentar (quanto maior, mais difícil ganhar slots)	↓*	↓	↑*
dec_bound	abaixo deste limite, o encontro pode diminuir (quanto menor, mais difícil liberar slots)	↑*	↑	↑*
inc_rate	taxa de acréscimo de slots	↑	↑	↓*
dec_rate	taxa de decréscimo de slots	↓	↓	↑*
max_dur	duração máxima de um encontro (quanto menor, mais tempo a ponte pode permanecer em HOLD)	↑*	↑	↓*
min_dur	duração mínima de um encontro (um valor pequeno permite maior redistribuição de slots)	↓*	↑*	↓
skip_pico	permite que a ponte não escalone uma picorrede em um turno (1=função desabilitada)	0	0*	1*
early_exit	permite a saída antecipada da ponte (um valor alto impede a saída antecipada)	↑	↑*	↓*

'*' indica os parâmetros de maior influência em cada métrica.
'↑' significa que o aumento do parâmetro melhora o desempenho da métrica.
'↓' significa que a diminuição do parâmetro melhora o desempenho da métrica.

Tabela 5.9: Parametrização do AISA.

O algoritmo AISA foi avaliado em três cenários, cada um visando a otimização de uma métrica de desempenho distinta. Após diversas simulações com parametrizações diferentes para o AISA, isolaram-se os parâmetros de maior influência em cada métrica. Apesar de o valor absoluto de cada parâmetro ser dependente da topologia, apresentou-

se, para cada métrica, um idéia geral sobre como os mesmos devem ser configurados. Esta idéia é resumida na Tabela 5.9.

O próximo capítulo apresenta as conclusões da dissertação e apresenta sugestões para a continuação deste trabalho.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho apresentou três contribuições principais: a proposta de um algoritmo de escalonamento intrapicorrede, a proposta de um algoritmo de escalonamento interpicorrede e uma ferramenta de simulação para a tecnologia Bluetooth. A próxima seção apresenta as conclusões do trabalho, seguida da seção contendo as sugestões de propostas para a continuação desta pesquisa.

6.1 Conclusões

O escalonamento intrapicorrede é a forma como o mestre de uma picorrede controla o acesso dos escravos à rede. Um escravo somente tem o direito de transmitir quando é escalonado pelo mestre. Por isso, a política de escalonamento adotada pelo mestre tem grande influência no desempenho da picorrede, em termos de vazão agregada e retardo dos pacotes.

A especificação Bluetooth define dois tipos de enlace entre estações: o enlace sem conexão (ACL), compartilhado entre o mestre e os escravos e o enlace orientado à conexão (SCO), criado individualmente entre o mestre e um escravo. No primeiro, trafegam os dados de melhor esforço e o algoritmo de escalonamento intrapicorrede é o responsável por tentar manter o canal sempre ocupado, aumentando a vazão agregada. O segundo tipo de enlace foi criado para fornecer vazão constante e retardo limitado para certos tipos de tráfego como, por exemplo, a voz. A existência de um ou mais enlaces SCO junto ao enlace ACL degrada fortemente o desempenho deste último.

O algoritmo de escalonamento intrapicorrede proposto, chamado DRR-CoS, visa permitir a coexistência, em um enlace ACL, de tráfego de melhor esforço e tráfego

com requisito de retardo limitado, atendendo à necessidade deste, sem degradar o desempenho daquele. Até onde se tem conhecimento, este compartilhamento do enlace ACL por mais de uma categoria de tráfego é um assunto pouco explorado em Bluetooth. O DRR-CoS baseia-se no *deficit round robin* (DRR), adaptado para as características do Bluetooth e acrescido de um parâmetro com o objetivo de limitar o intervalo entre escalonamentos sucessivos das estações, cujo tráfego apresente restrição de retardo. Este parâmetro foi denominado **IPol** (intervalo de *polling*).

As simulações realizadas com o algoritmo DRR-CoS mostraram ser plenamente viável o compartilhamento do enlace ACL entre o tráfego de melhor esforço e o tráfego com requisito de retardo limitado. O retardo deste último foi mantido dentro do limite estabelecido pelo parâmetro **IPol**, ao passo que a vazão agregada do tráfego de melhor esforço foi cerca de 30% superior ao máximo teórico permitido para o enlace ACL na presença de um enlace SCO.

O escalonamento interpicorrede é a forma como uma estação divide o seu tempo para participar de múltiplas picorredes. Normalmente, as estações dispõem de apenas uma interface Bluetooth, impedindo-as de participar de múltiplas picorredes simultaneamente. A estação que participa de mais de uma picorrede é chamada de ponte, devido à sua capacidade de encaminhar pacotes de uma picorrede para outra. A ponte aumenta o alcance da rede, a qual passa a ser chamada de *scatternet*.

O algoritmo usado pela ponte para organizar sua participação nas picorredes influi diretamente na vazão e no retardo dos fluxos que a atravessam ou a esta se destinam. Além disso, as pontes, juntamente com os mestres das picorredes, costumam ser as estações que mais consomem energia. Portanto, o algoritmo de escalonamento usado pelas pontes também afeta o consumo de energia das mesmas.

Foi feito um estudo detalhado de diversas propostas de escalonamento interpicorrede existentes. A partir das similaridades entre as propostas, criou-se uma classificação para os algoritmos, composta de duas categorias: políticas de escalonamento com decisão isolada e com decisão distribuída. Nos algoritmos com decisão isolada, todas as decisões sobre o escalonamento interpicorrede são tomadas exclusivamente pela ponte, enquanto nos algoritmos com decisão distribuída, as decisões são tomadas após acordo entre a ponte e seus vizinhos.

Antes de ser iniciada a implementação do algoritmo interpicorrede proposto, foram definidos seus objetivos. O algoritmo deve permitir a adaptação da ponte às condições do tráfego vigente e deve ser parametrizável, possibilitando que uma determinada métrica de desempenho seja priorizada. As métricas em questão são: a vazão agregada do tráfego, o retardo dos pacotes e o consumo de energia. Além dos objetivos estabelecidos, o algoritmo também deve possuir as seguintes características de implementação: alterar o mínimo possível a especificação Bluetooth e evitar a criação de sinalização adicional entre estações. Desta forma, a implementação da proposta torna-se mais simples e com mais chance de ser incorporada à especificação Bluetooth.

O resultado dos estudos foi a proposta de escalonamento interpicorrede denominada *Adaptive Interpiconet Scheduling Algorithm* (AISA). A proposta AISA se encaixa na categoria dos algoritmos com decisão isolada.

Para testar o AISA, foram definidos três cenários, explorando as três métricas de desempenho citadas. Uma das dificuldades encontradas foi a escolha dos parâmetros em cada cenário, já que existem nove parâmetros configuráveis. Após diversas simulações, foi possível isolar os parâmetros de maior influência em cada uma das métricas de desempenho. Apesar do valor absoluto de cada parâmetro ser dependente da topologia, apresentou-se, para cada métrica, um idéia geral sobre como os mesmos devem ser configurados.

O algoritmo AISA teve bom desempenho com as três métricas, sobretudo no caso da economia de energia. Por bom desempenho, entende-se a otimização da métrica de desempenho definida como prioritária em um cenário, com o mínimo de prejuízo para as demais métricas. Os resultados servem como um indicativo da possibilidade do AISA ser empregado, com sucesso, em variados tipos de cenários. Dentre os trabalhos pesquisados, não se tem conhecimento de outra proposta que busque atender, com um único algoritmo, a cenários com requisitos diferentes.

Todas as simulações foram realizadas com a ferramenta BlueNetS (*Bluetooth Network Simulator*), desenvolvida ao longo deste trabalho. A implementação da ferramenta de simulação BlueNetS ocupou uma grande parcela do tempo desta pesquisa e seu desenvolvimento foi motivado pela não existência, no início deste trabalho, de uma ferramenta adequada para a simulação dos algoritmos propostos. A única ferramenta de código

aberto disponível na época era o BlueHoc. Entretanto, o BlueHoc apresentava uma série de limitações, principalmente no referente às *scatternets*, das quais havia poucas funcionalidades implementadas. Esta e outras razões levaram ao desenvolvimento do BlueNetS.

As seguintes funcionalidades estão implementadas na ferramenta: duplexação por divisão no tempo (TDD), configuração de enlaces ACL, encaminhamento de pacotes nos sentidos mestre-escravo e escravo-mestre, mecanismos de correção de erros (FEC) e retransmissão (ARQ), segmentação e remontagem (SAR), modelo de erros provocados pela distância entre as estações, funcionamento da ponte nas configurações escravo/escravo e mestre/escravo e diversos algoritmos de escalonamento intrapicorrede e interpicorrede.

As funcionalidades citadas permitem o estudo do tráfego de dados entre as estações. Entretanto, a sinalização relacionada à descoberta de dispositivos e à fase de conexão não foi implementada. No estágio atual, BlueNetS pode ser empregado em estudos que não dependam da fase de estabelecimento de conexões. Neste trabalho, foi priorizado o estudo de políticas de escalonamento. Entretanto, existem muitas outras aplicabilidades para a ferramenta. Por exemplo, pode-se utilizá-la em simulações para o estudo de protocolos de roteamento, de protocolos de descoberta de serviços, do comportamento do protocolo TCP, da interferência entre picorredes, entre outros.

A ferramenta BlueNetS é uma extensão ao conhecido simulador de redes NS-2. Sua implementação procurou modificar o mínimo possível o código compilado, em linguagem C++, do simulador original. O resultado foi a criação de dois novos módulos e a modificação de apenas dois existentes. A parte do código relacionada às políticas de escalonamento foi deixada na linguagem interpretada OTcl. Como as políticas de escalonamento eram constantemente alteradas, a linguagem OTcl evitou muitas recompilações de código.

6.2 Trabalhos Futuros

Como trabalhos futuros, são feitas as seguintes sugestões.

Quanto aos resultados das simulações, foi estudado, em detalhes, um cenário para cada métrica de desempenho. No entanto, existem mais cenários onde a tecnologia Bluetooth é aplicável. Para se obter resultados ainda mais precisos sobre a parametrização do mecanismo AISA para cada métrica de desempenho, devem ser estudados outros cenários. Posteriormente, devem ser feitas análises comparativas entre os diversos cenários de cada métrica, de forma a destacar os parâmetros comuns a todos.

Não foram testados modelos específicos de tráfego de áudio, de vídeo e de navegação na Internet (tráfego Web). Estas categorias de tráfego multimídia introduzem outras métricas de desempenho como variação do retardo e tempo de resposta de uma solicitação, merecendo uma pesquisa futura.

Conforme são adicionadas novas categorias de tráfego em uma *scatternet*, aumenta-se a necessidade da criação de classes de tráfego, fornecendo diferentes garantias para as diferentes classes. O fornecimento de qualidade de serviço ao tráfego deve envolver tanto a política de escalonamento intrapicorrede quanto a interpicorrede.

No caso do escalonamento intrapicorrede, o mestre deve escalonar os escravos de acordo com a prioridade dos seus fluxos. A proposta DRR-CoS separa duas categorias: os fluxos com requisito de retardo limitado e os fluxos de melhor esforço. Não é possível distinguir entre dois fluxos com necessidades de retardo ou entre dois fluxos com necessidades de banda diferentes. O primeiro caso pode ser resolvido com a criação de mais de um valor IPol (parâmetro que limita o retardo) pelo DRR-CoS. O segundo caso envolve maiores modificações no código para permitir a implementação de uma política de escalonamento hierárquico.

O DRR-CoS não foi comparado com políticas de escalonamento com restrições temporais, como o Earliest Deadline First (EDF). Assim como o DRR-CoS, os algoritmos dessa categoria apresentam a preocupação de não ultrapassar um valor de retardo máximo estipulado. Por isso, a implementação de algoritmos dessa modalidade é sugerida como forma de apontar a política que melhor se adapta às características do Bluetooth.

No caso do escalonamento interpicorrede, a distinção entre várias classes de tráfego é mais complexa. O mecanismo AISA foi criado para permitir que a ponte melhore o

desempenho de uma determinada categoria de tráfego, de uma forma simples, evitando negociações entre as estações. Certamente, para fornecer garantias diferenciadas aos fluxos, serão necessárias alterações no AISA, inclusive com o acréscimo de mensagens de sinalização trocadas entre a ponte e seus vizinhos.

O algoritmo AISA forneceu resultados muito bons em termos de economia de energia. Estes resultados apontam para uma possível aplicação do AISA no estudo de redes de sensores. Essas redes, sozinhas, constituem uma área de estudo que vem sendo bastante explorada. Entretanto, não foram encontrados, durante esta pesquisa, trabalhos específicos de redes de sensores com a tecnologia Bluetooth. Sugere-se, como trabalho futuro, o estudo da viabilidade da formação de redes de sensores com Bluetooth, mais especificamente, utilizando o AISA como política de escalonamento.

Vários protocolos de redes podem ser estudados, criando-se os módulos necessários no simulador ns-2. Como o BlueNetS é baseado no código do ns-2, o usuário não precisa ter um conhecimento profundo sobre a implementação do BlueNetS para criar, por exemplo, um módulo de roteamento para redes *ad hoc*, que possa ser aplicado sobre Bluetooth. Basta o usuário conectá-lo ao BlueNetS como ele o faz com um módulo de roteamento já existente no ns-2.

Quanto à ferramenta BlueNetS em si, alguns aperfeiçoamentos podem ser efetuados. Além do modelo de erros existente, podem ser incorporados modelos de interferência entre picorredes e de interferência causada por fontes externas como IEEE 802.11. A interferência entre picorredes tem bastante influência no comportamento do tráfego quando o número de picorredes cresce. Apesar do BlueNetS não simular a fase de conexão, a ferramenta pode simular o tempo gasto pelas estações, quando elas saem temporariamente de sua picorrede para procurar novas estações. Finalmente, pode ser feito o transporte do código em OTcl para a linguagem C++, melhorando um pouco o desempenho do simulador. Essa tradução será bastante trabalhosa, uma vez que o código em OTcl, contendo a configuração da *scatternet* e os algoritmos de escalonamento, soma mais de 1000 linhas.

Referências Bibliográficas

[1] IEEE Standards Association. **802.11 Working Group for Wireless Local Area Networks**. *http://grouper.ieee.org/groups/802/11*. arquivo consultado em outubro de 2002.

[2] The European Telecommunications Standards Institute (ETSI). **High performance radio local area networks**. *http://www.etsi.org/technicalfocus/home.htm*. arquivo consultado em dezembro de 2001.

[3] Home Radio Frequency Working Group. **HomeRF Shared Wireless Access Protocol (SWAP)**. *http://www.homerf.org*. arquivo consultado em dezembro de 2001.

[4] Bluetooth Special Interest Group. *http://www.bluetooth.com*. arquivo consultado em maio de 2002.

[5] Haartsen, J. **The Bluetooth Radio System**. *IEEE Personal Communications 7*, 1, fevereiro de 2000, 28–36.

[6] IEEE Standards Association. **802.15 WPAN Task Group 1 (TG1)**. *http://www.ieee802.org/15/pub/TG1.html*. arquivo consultado em outubro de 2002.

[7] Liu, Z., Nain, P., and Towsley, D. **On Optimal Polling Policies**. *Queueing Systems 11*, 1992, 59–83.

[8] Yaiz, R. A., and Heijenk, G. **Polling Best Effort Traffic in Bluetooth**. *Proceedings of 4th International Symposium on Wireless Personal Multimedia Communications (WPMC 2001)*, setembro de 2001.

[9] Capone, A., Gerla, M., and Kapoor, R. **Efficient Polling Schemes for Bluetooth Piconets**. *IEEE ICC 2001*, junho de 2001.

[10] Das, A., Ghose, A., Razdan, A., Saran, H., and Shorey, R. **Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless ad-hoc Network**. *Proceedings of IEEE Infocom*, abril de 2001.

[11] Kalia, M., Bansal, D., and Shorey, R. **Data Scheduling and SAR for Bluetooth MAC**. *IEEE Vehicular Technology Conference (VTC)*, maio de 2000.

[12] Baatz, S., Frank, M., Kühl, C., Martini, P., and Scholz, C. **Adaptive Scatternet Support for Bluetooth using Sniff Mode**. *Proceedings of the 26th Annual Conference on Local Computer Networks*, novembro de 2001.

[13] Johansson, P., Kapoor, R., Kazantzidis, M., and Gerla, M. **Rendezvous Scheduling for Bluetooth Scatternets**. *Proceedings of ICC 2002*, abril de 2002.

[14] Racz, A., Miklos, G., Kubinszky, F., and Valko, A. **A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets**. *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, outubro de 2001.

[15] Zhang, W., and Cao, G. **A Flexible Scatternet-wide Scheduling Algorithm for Bluetooth Networks**. *IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2002.

[16] Har-Shai, L., Kofman, R., Zussman, G., and Segall, A. **Inter-Piconet Scheduling in Bluetooth Scatternets**. *Proceedings of the OPNETWORK 2002 Conference*, agosto de 2002.

[17] Priess, W., Rezende, J. F., Pirmez, L., and Carmo, L. F. R. C. **BlueNetS: Uma Ferramenta para Estudo do Desempenho do Bluetooth**. *IV Workshop de Comunicação Sem Fio e Computação Móvel (WCSF2002)*, outubro de 2002.

[18] **The Network Simulator (ns-2)**. *http://www.isi.edu/nsnam/ns*. arquivo consultado em novembro de 2002.

[19] Soares, L. F. G., Lemos, G., and Colcher, S. **Redes de Computadores: Das LANs, MANs e WANs às Redes ATM**. *Editora Campus*, 1995.

[20] Weiser, M. **Some Computer Science Problems in Ubiquitous Computing**. *Communications of the ACM 36*, 7, julho de 1993, 75–84.

[21] Internet Engineering Task Force (IETF). **Mobile Ad Hoc Networks (MANET) Working Group**. *http://www.ietf.org/html.charters/manet-charter.html*. arquivo consultado em maio de 2002.

[22] The International Telecommunication Union. **Open Distributed Processing - Reference Model: Foundations**. *ISO 10746-2/ITU-T X.902 Information technology*, novembro de 1995.

[23] Vogel, A., Kerherve, B., von Bochmann, G., and Gecsei, J. **Distributed Multimedia and QoS: a Survey**. *IEEE Multimedia 2*, 2, 1995, 10–19.

[24] Chalmers, D., and Sloman, M. **A Survey of Quality of Service in Mobile Computing Environments**. *IEEE Communications Survey 2*, 2, 1999.

[25] Blair, G., and Stefani, J.-B. **Open Distributed Processing and Multimedia**. *Addison-Wesley*, 1997.

[26] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. **An Architecture for Differentiated Services**. *RFC 2475*, dezembro de 1998.

[27] Wroclawski, J. **The Use of RSVP with IETF Integrated Services**. *RFC 2210*, setembro de 1997.

[28] Chakrabarti, S., and Mishra, A. **QoS Issues in Ad Hoc Wireless Networks**. *IEEE Communication Magazine*, fevereiro de 2001.

[29] Vaidya, N. **Open Problems in Mobile Ad Hoc Networking**. *Workshop on Wireless Local Networks in the 26th IEEE Conference on Local Computer Networks*, novembro de 2001.

[30] Johansson, N., Korner, U., and Johansson, P. **Performance Evaluation of Scheduling Algorithms for Bluetooth**. *Broadband Communications: Convergence of Network Technologies*, 2000, 139–150. Kluwer Academic Publishers.

[31] Zürbes, S., Stahl, W., and Matheus, K. **Radio Network Performance of Bluetooth**. *IEEE ICC 2000 3*, 2000, 1563–1567.

[32] Golmie, N., Dyck, R. E. V., and Soltanian, A. **Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation**. *Proc. of the 4th International ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, julho de 2001.

[33] **Bluetooth Core Specification v. 1.1**. *Part B, Section 10.9, http://www.bluetooth.com*, fevereiro de 2001, 120.

[34] Bluetooth Special Interest Group. **Personal Area Networking Profile (PAN) v0.95a**. *http://www.bluetooth.com*.

[35] Bhagwat, P., and Rao, S. P. **On the characterization of Bluetooth scatternet topologies**. *Mobihoc*, agosto de 2000.

[36] Miklos, O., Rácz, A., Valkó, A., and Johansson, P. **Performance Aspects of Bluetooth Scatternet Formation**. *First Annual Workshop on Mobile and Ad Hoc Networking and Computing (MobiHoc)*, agosto de 2000.

[37] Salonidis, T., Bhagwat, P., Tassiulas, L., and LaMaire, R. **Distributed Topology Construction of Bluetooth Personal Area Networks**. *IEEE Infocom*, abril de 2001.

[38] Ramachandran, L., Kapoor, M., Sarkar, A., and Aggarwal, A. **Clustering Algorithms for Wireless Ad Hoc Networks**. *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, agosto de 2000.

[39] Law, C., Mehta, A., and Siu, K.-Y. **Performance of a new Bluetooth Scatternet Formation Protocol**. *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing*, outubro de 2001.

[40] Záruba, G. V., Basagni, S., and Chlamtac, I. **Bluetrees - Scatternet Formation to Enable Bluetooth-based Ad Hoc Networks**. *Proceedings of IEEE ICC 2001 1*, junho de 2001, 273–277.

[41] Tan, G. **Self-organizing Bluetooth Scatternets**. *Master Thesis*, janeiro de 2002.

[42] Bluetooth Special Interest Group. **Bluetooth Network Encapsulation Protocol (BNEP) v0.95a**. *http://www.bluetooth.com*.

[43] Johansson, P., Kapoor, R., Gerla, M., and Kazantzidis, M. **Bluetooth: an Enabler of Personal Area Networking**. *IEEE Network, Special Issue in Personal Area Networks*, outubro de 2001.

[44] Raman, B., Bhagwat, P., and Seshan, S. **Arguments for Cross-Layer Optimizations in Bluetooth Scatternets**. *Symposium on Applications and the Internet (SAINT 2001)*, janeiro de 2001.

[45] Atwal, K., and Akers, R. **Transmission of IP Packets over Bluetooth Networks**. *IETF Draft*, maio de 2001.

[46] Bhagwat, P., and Segall, A. **A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets**. *The Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC 1999)*, novembro de 1999.

[47] Nagle, J. **On Packet Switches with Infinite Storage**. *IEEE Transactions on Communications 35*, 4, abril de 1987, 435–438.

[48] Shreedhar, M., and G.Varghese. **Efficient Fair Queueing Using Deficit Round-Robin**. *Proceedings of the ACM Sigcomm*, setembro de 1995.

[49] Hahne, E. L. **Round Robin Scheduling for Fair Flow Control in Data Communication Networks**. *Phd Thesis*, dezembro de 1986.

[50] Johansson, P., Johansson, N., Körner, U., Elgg, J., and Svennarp, G. **Short Range Radio Based Ad Hoc Networking: Performance and Properties**. *Proceedings of ICC 1999*, 1999.

[51] Johansson, N., Alriksson, F., and Jönsson, U. **JUMP Mode - A Dynamic Window-based Scheduling Framework for Bluetooth Scatternets**. *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, outubro de 2001.

[52] Markopoulou, A., Tobagi, F., and Karam, M. **Assessment of VoIP Quality over Internet Backbones**. *IEEE Infocom*, junho de 2002.

[53] IBM Corporation. **BlueHoc: Bluetooth Performance Evaluation Tool**. *http://www-124.ibm.com/developerworks/opensource/bluehoc*. arquivo consultado em setembro de 2002.

[54] Kumar, A., and Gupta, R. **Capacity Evaluation of Frequency Hopping Based Ad-Hoc Systems**. *ACM Sigmetrics/Performance*, junho de 2001.

[55] Schwartz, M. **Broadband Integrated Networks**. *Prentice Hall Inc*, 1996.

[56] Brandy, P. **A Technique for Investigating On/Off Patterns of Speech**. *Bell Labs Tech. Journal 44*, 1, janeiro de 1965, 1–22.

[57] Liu, C. L., and J.W.Layland. **Scheduling Algorithms for Multiprogramming in a Hard-Real-Time**. *Journal of the ACM 20*, 1, janeiro de 1973, 46–61.

[58] Philips Semiconductors. **UAA 3558 Bluetooth RF Transceiver**. *http://www.semiconductors.philips.com/technologies/bluetooth*. arquivo consultado em setembro de 2002.

[59] Ericsson Microeletronics. **PBA 313 01/3 Bluetooth Radio**. *http://www.ericsson.com/microe/products/bluetooth_solutions*. arquivo consultado em setembro de 2002.

[60] Bluetronics. **IRMBB2 Bluetooth Radio Module**. *http://www.bluetronics.com*. arquivo consultado em setembro de 2002.

[61] Silicon Wave. **SiW1701 Bluetooth Radio Modem**. *http://www.siliconwave.com/bluetooth.html*. arquivo consultado em setembro de 2002.

[62] Linsky, J. **Bluetooth and Power Consumption: Issues and Answers**. *RF Design Magazine*, novembro de 2001, 74–77, 94–95.

[63] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. **Wireless Sensor Networks: a Survey**. *Computer Networks: The International*

Journal of Computer and Telecommunications Networking 38, 4, março de 2002, 393–422.

[64] Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M. B. **Energy-Aware Wireless Microsensor Networks**. *IEEE Signal Processing Magazine 19*, 2, março de 2002, 40–50.